情報処理学会論文誌 Vol.54 No.7 1930–1936 (July 2013)

推薦論文

局面評価関数を使う新たなUCT探索法の提案とオセロによる評価

橋本 剛[1,a)] 前原 彰太[2] 川島 哲哉[3] 小林 康幸[4]

受付日 2012年11月30日, 採録日 2013年4月5日

概要：新たなゲーム木探索法としてモンテカルロ木探索，特にUCTが成功を収め広く研究されている．だが主なターゲットであるコンピュータ囲碁では局面評価関数の計算が困難であるため，パターンマッチングなどにより手の評価値を計算し強いプレイヤに近いモンテカルロシミュレーションを目指す研究がさかんに行われているものの，局面評価関数を使ったUCTの研究はこれまでほとんどなかった．本研究では新たに局面評価値をUCB値に加える手法，UCT+を提案する．初めて探索する局面で評価関数の値が高い子局面を優先的に探索するので，性能の良い評価関数があれば簡単にUCTを強くすることが期待される．実験ではすでに優れた局面評価関数が存在するオセロに提案手法を実装し評価を行った．評価関数はオープンソースで世界最強プログラムZebraのものを使った．その結果，提案手法はUCTに対し圧倒的な性能を示しその有効性が実証された．

キーワード：UCT+，オセロ，モンテカルロ木探索，評価関数

A New UCT Search Method Using Position Evaluation Function and its Evaluation by Othello

Tsuyoshi Hashimoto[1,a)] Shouta Maehara[2] Tetsuya Kawashima[3]
Yasuyuki Kobayashi[4]

Received: November 30, 2012, Accepted: April 5, 2013

Abstract: The Monte Carlo tree search, particularly UCT, is extensively studied as a new game tree search method. Study of evaluation function on UCT is mainly focused on move evaluation functions such as using pattern matching. However, UCT using position evaluation function has not been studied because of the difficulty in calculating position evaluation function in the game of GO, which is the main target of UCT research. We propose a new method UCT+ that adds position evaluation values to the UCB value. This method is expected easily to make UCT strong in case of existing good position evaluation function, as it gives priority to child positions of high evalation value that has not searched yet. Experiments are performed using the game of Othello, that already has strong position evaluation functions. Evaluation function of the Zebra, the strongest open source othello program, is used for the experiments. The results show the overwhelming ability of proposed method and its effectiveness is verified.

Keywords: UCT+, Othello, Monte Carlo tree search, evaluation function

1. はじめに

2人零和完全情報ゲームはミニマックス探索と評価関数を組み合わせるのが一般的である．オセロのトッププログラムはパターンの学習による評価関数を用いている [1]

本論文の内容は2011年10月の電気・情報関連学会中国支部連合大会にて報告され，支部長により情報処理学会論文誌ジャーナルへの掲載が推薦された論文である．

[1] 松江工業高等専門学校情報工学科
Department of Information Engineering, Matsue College of Technology, Matsue, Shimane 690–8518, Japan
[2] アイティフォー
IT FOR Inc., Chiyoda, Tokyo 102–0082, Japan
[3] リゾーム
RHIZOME CO., LTD., Okayama 701–0165, Japan
[4] 島根大学大学院総合理工学研究科
Graduate School of Science and Engineering, University of Shimane, Matsue, Shimane 690–8504, Japan
[a)] hashimoto@matsue-ct.jp

が，探索手法はミニマックス法を基本としている．Buro の Logistello が 1997 年に当時の世界チャンピオンを破る [2] など，オセロプログラムはすでに人間では勝てない強さである．チェスや将棋でもミニマックス探索を用いるのが一般的だが，囲碁では評価関数の作成が難しく，コンピュータ囲碁の作成は難しいとされてきた．だが，近年コンピュータ囲碁ではモンテカルロ木探索 [3] が 2006 年に Computer Olympiad で優勝して以降注目を集め，大流行している．モンテカルロ法と呼ばれる乱数を用いたプレイヤ同士で終局までゲームを行い，その結果を局面の評価としてゲーム木探索と組み合わせたものがモンテカルロ木探索である．代表的なものとして UCT [4] がある．UCT は UCB [5] という値が高いノードに多くの探索を割り当て，多く探索されたノードだけを展開する手法である．

囲碁やオセロはプレイヤがお互いにランダムに手を選択しても一定の手数以内に終局しやすい．モンテカルロ法は終局までランダムに手を選び勝率を計算するので，このようなゲームに適していると考えられる．一方，将棋では王を詰ます，駒得を目指すなどの目的を持った手がランダムで選ばれないと収束しにくいモンテカルロ法で強くすることは難しいことが知られている [6], [7], [8]．またモンテカルロ法はルール以外のゲームに関する知識が不要な手法である．そのため，ヒューリスティックによる評価関数が作りにくいコンピュータ囲碁において特に有効な手法として注目され，研究がさかんに行われている [3], [4], [9]．近年ではパターンマッチングなどにより手の評価値を計算し，純粋なランダムではなく強いプレイヤに近いモンテカルロシミュレーション（以下プレイアウトとする）を実現することで強化を図ることがさかんに行われている [9], [10], [11]．一方，囲碁以外のゲームで UCT を使った研究も現在はさかんで，Amazons [12], [13] や LOA（Lines of Action）[14] など 2 人零和完全情報ゲームをはじめ，Nested Monte-Carlo 探索を使ったパズル Morpion Solitaire の研究 [15], [16] など多岐にわたっている．だが，モンテカルロ木探索の主なターゲットが囲碁であるために，手ではなく局面の評価関数を使う研究はほとんど行われていない．

本稿では局面評価関数を使った UCT の性能向上について論じる．UCB 値に局面評価関数を加えることで UCT の性能を大幅に向上させる手法を提案し，コンピュータオセロで実験を行う．

2 章では本研究の基本となる UCB 値とモンテカルロ木探索について簡単に述べる．3 章では関連研究としてモンテカルロ木探索と評価値を用いる研究を紹介する．4 章では本研究の焦点である提案手法について述べる．5 章ではオセロへの提案手法の実装について述べる．オセロはパスを除くと必ず 60 手以内に終局するので，モンテカルロ法に適した題材と予想されるが，すでに人間より強いためにモンテカルロ法に関連する研究は少ない．6 章では世界最高峰のオセロプログラムである zebra [17] で使用されている局面評価関数を実装した実験について述べる．7 章では提案手法と，他のモンテカルロ木探索と評価値を用いる研究との性能を比較し，提案手法の位置づけを示す．8 章では新しい UCT 探索法としての有効性について考察を述べる．9 章では今後の課題や展望について述べる．

2. UCB 値とモンテカルロ木探索

ある局面 P の UCB（Upper Confidence Bound）[5] 値は以下のように表される．

$$\overline{X_i} + c\sqrt{\frac{2\log n}{n_i}} \tag{1}$$

局面 P の手 i に n_i 回のプレイアウトが行われたときの勝率を $\overline{X_i}$，n を局面 P で行われたプレイアウト総数，c を適当に定めた定数とする．

この式の構成を（勝率項）＋（バイアス項）と呼ぶことにする．この式では，勝率項が高いノードにプレイアウトを多く割り当てつつ，プレイアウト数が少ないノードを探索する必要性を考慮して，探索数の少ないノードほどバイアス項が高くなる．

UCB 値にはいくつかの種類があり，上記の値は正確には UCB1 値 [5] だが，本稿では式 (1) を UCB 値として進めていく．

モンテカルロ法はすべての手を乱数で選択する．しかし，悪い手は結局選ばれないので，できるだけ探索しない方が効率良い．UCB 値を用いることで，計算量を小さく保ちつつ，良さそうな手に多くの探索を割り当てることができる．

モンテカルロ法に木探索を組み合わせた手法がモンテカルロ木探索である．囲碁などの 2 人零和完全情報ゲームでは，ミニマックス木探索で探索末端の評価値をモンテカルロプレイアウトによる勝率としたものをモンテカルロ木探索と呼ぶ．

モンテカルロ木探索に UCB 値を用いてノードを選んでいく手法が UCT（UCB applied to Trees）[4] である．UCT 探索では UCB 値の高いノードが多く選ばれ，定めた閾値を超えるとノードが展開される．

3. 関連研究

モンテカルロ木探索に評価関数を用いる関連研究を紹介する．

主流は囲碁を中心とした手の評価関数の研究で，プレイアウトで高い評価の手を選ばれやすくし精度を高めることが目的である．評価項目としては着手を中心とした 8 近傍のパターンなどが使われることが多い．最近は評価値の機械学習による計算が主流で，囲碁を中心にさかんに研究されている [10], [11]．Amazons でも同様の研究が行われて成果をあげている [13]．手の評価関数でプレイアウトだけ

でなく UCT そのものの動きを制御することも行われている [11]．囲碁では局面評価関数の作成が難しいため古典的なミニマックス探索からモンテカルロ探索へと移行した経緯があるため，現在は手の評価関数の研究がほとんどである．

囲碁以外ではモンテカルロ木探索で局面評価関数を使う研究も存在する．文献 [12] では，ミニマックス探索ベースの Amazons プログラム開発者であった著者がモンテカルロ木探索に既存の評価関数を使っている．その使い方は，終局までプレイアウトを行わず一定の深さに達したら勝ち負け判定の代わりに評価関数を呼び出し，評価値が相手より高いプレイヤを勝ちと見なしており，終局までプレイアウトを行う場合に比べて有意に強くなると報告されている．同様の議論はモンテカルロ将棋でも行われている [6], [7]．

以上のようにモンテカルロ木探索に評価関数を使う研究は多く存在するものの，UCB 値に直接評価関数を使う研究は行われていない．

4. 提案手法

UCT 探索では UCB 値の高い手が選択されるが，UCB 値は一般にプレイアウトで得られる勝率が支配的である．しかし，勝率だけではプレイアウト数を多くしないと評価値の誤差が大きく，良い結果が得られるまでに時間がかかってしまう．ある程度局面の良し悪しを判断できる評価関数がある場合には，この問題を大きく改善できる可能性がある．従来は関連研究で述べたように評価関数を UCB 値とは別に用いる手法が多く考案されている．プレイアウトの質を高めるために評価関数が用いられてはいるが，選択するノードの比較には UCB 値が用いられている．

本稿では UCB の勝率項に適度にスケーリングした評価関数を組み合わせる方法を提案し，この評価値を UCB+と名付ける．そして，UCB+を最大化するノードを選択していくモンテカルロ木探索を UCT+とする．

局面 P で i 番目の子局面が適当にスケーリングされた局面評価値 E_i を持ち，n_i 回プレイアウトが行われたとする．E_i を n_i が大きくなるにつれて単調に減少させる関数 $E_i(n_i)$ を式 (1) の評価関数に加え，従来の UCB 値の代わりとする．式で表すと次のようになる．

$$\overline{X_i} + E_i(n_i) + c\sqrt{\frac{2\log n}{n_i}} \tag{2}$$

局面評価値 E_i を n_i が大きくなるにつれて減少させる理由は，プレイアウト試行回数 n_i が大きくなるほどヒューリスティックによる局面評価値より勝率を重視すべきであると考えるからである．局面評価値 E_i を勝率項にただ足すだけだと，n_i が大きくなっても E_i はそのまま残ってしまう．本稿の実装では以下のように計算した．

$$E_i(n_i) = \frac{E_i}{\sqrt{n_i}} \tag{3}$$

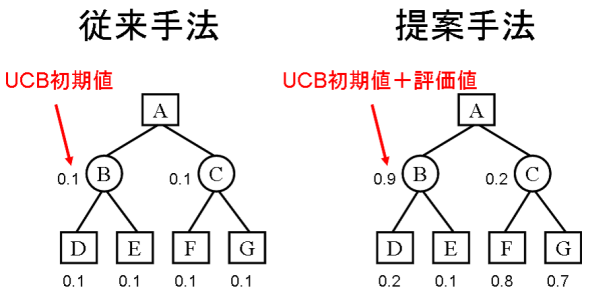

図 1 未探索局面 UCB 値：従来手法と提案手法の違い

Fig. 1 UCB values on initial positions: difference between the existing method and the proposed method.

UCT（従来手法）と UCT+（提案手法）による違いは図 1 のようになる．従来の UCT では，未探索局面の UCB 値が同じであるため，各ノードを順番に同じように探索しないとならない．プレイアウト数を多くしないと誤差が大きく，ある程度精度の高い結果を得るには時間がかかってしまう．提案手法 UCT+では，未探索局面でも局面評価値によって UCB+値が変わってくるので，局面評価値上良いと思われる局面の探索が早く行われる（図ではノード B, F, G の順）．局面評価値の性能が良く，ある程度勝率との相関が高ければ，従来手法より誤差の収束が早く，短時間である程度精度の高い結果を得られると予想される．

UCT+は式 (1) に直接，局面評価関数を加えて探索を行うという点で，今までの手法とは大きく異なる．

5. オセロによる実装

5.1 オセロの評価関数

本稿ではオセロを用いて実験を行う．オセロはパスを除いて必ず一定の手数（60 手）以内で収束するので，モンテカルロ木探索研究の題材として適していると考えた．実際にヒューリスティックスを用いないピュアな UCT でオセロプログラムを作ったところ，簡単に中級者程度の強さになった．またすでに多くの優れた局面評価関数が知られていることより，本研究の題材に特に適していると思われる．

本稿ではフリーソフトで世界最高峰の強さを持つ Zebra で用いられている局面評価関数を使う．

5.2 オセロによる予備実験

UCT+が効果を発揮するためには，扱う変数に適当な係数を与えなければならない．オセロの評価関数として Zebra を用いるとしたが，そのまま式 (2) の評価関数に組み込んでも良い結果は得られない．本稿では実験により適当な係数を算出した．

本稿では次章で述べる棋譜ファイルの 900 局面約 8,000 個のノードについて判別分析を行うことで得られた判別関数の係数の比率を基準にして，i 番目のノードで得られる評価値である *zebravalue* を用いた局面評価関数とバイアス項の係数 c を次のように定めた．

$$E_i = \frac{zebravalue_i}{30}, c = 1$$

式 (2) の E_i と c は上記のように，扱うゲームと用いる変数によって適当にスケーリングする必要がある．

6. 実験

提案手法の有意性を評価する方法として，問題セットの正解率による比較と対戦実験による比較を行う．実験を行ったマシンは CPU Pentium4 3.2 GHz, メモリ 512 MB で，OS は Debian Linux 5.0.3，コンパイラは gcc である．

6.1 問題セットによる実験

あらかじめ完全探索によって最善手とその値が分かっている棋譜ファイルを用いる．これはオセロの棋譜から，調べたい問題局面において完全探索の値が勝ちであるノードと負けであるノードが必ず 1 つずつ以上含まれている局面だけを集めている．完全探索はオセロの終盤ソルバである scrzebra と Wzebra [17] によって行った．このプログラムは $\alpha\beta$ 法を基本としたアルゴリズムを用いて完全探索を行っている．

この実験で S は問題局面の手数を表すこととする．$S = 40$ と書いている場合，初手から 40 手目が終了した局面ということである．問題局面 $S = 40, 30, 20$ でそれぞれ 300 件ずつ用意した．例として，$S = 50$ の**図 2** では以下のように書き込まれている．

(g, 1), −1, (h, 1), 1, (h, 2), −1, (h, 3), 0, (h, 7), 1,

括弧の中が着手できる手の場所，値が終局での勝敗（1 なら勝ち，0 なら引分け，−1 なら負け）である．この例では次手は黒番であり，着手できる場所は 5 カ所ある．ファイルにはその着手できる場所に置いた場合から終局までお互いに最善手を打った場合の先手にとっての最終値が記載されている．つまり，先手が (2) の h1 か (5) の h7 を打った後，お互いに最善手を打てば先手の黒番が勝てることになる．

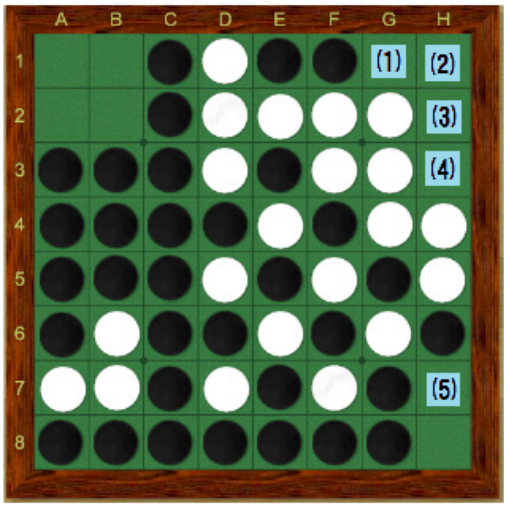

図 2 問題局面の例
Fig. 2 An example of a problem position.

選んだ手の完全探索結果ファイルに記載されている値が正なら正解の手，負なら不正解の手，0 なら引き分けの手として以下の式により正解率を求める．

正解率＝ (正解数＋引き分け数× 0.5) ÷棋譜数

なお，$S = 20$ では完全探索を行うのにあまりにも時間がかかりすぎるため，24 手読みの設定をした Wzebra を用いて各合法手を評価し，その値が正なら勝ち，負なら負けとして実験に用いている．

6.2 問題セット実験結果

各手法の実験結果を**表 1**，**表 2**，**表 3** に載せる．なお，表 2 の従来の UCB 値のみを用いた UCT では UCT+と比較するために c は UCT+と同じ値（$c = 1$）にしている．また，すべての表の探索時間は 300 件全体の合計，%は正解率である．

表 1 には比較のために別の手法による正解率を載せている．ランダムプレイは合法手の中から完全にランダムに 1 手選ばせたものである．MC（モンテカルロ）法は探索局面の深さ 1 の各ノードからそれぞれ 2,000 回ずつプレイアウトを行い，勝率値が最も高いノードを選択したものである．木探索でないためプレイアウトを多くしても正解率はほとんど変わらない．表 1 と表 2 を比較すると，S が大きくなり終局に近い問題ほど UCT の正解率が高くなっている．$S = 20$ では単純な手法と UCT の正解率がほとんど変わらないが，UCT ではこの程度のプレイアウト数だとラ

表 1 単純な手法の正解率
Table 1 Accuracy rate by simple methods.

	S=20	time	S=30	time	S=40	time
ランダム	33.2%	0.02 秒	45.3%	0.02 秒	46.7%	0.02 秒
MC 法	30.3%	913 秒	45.0%	413 秒	57.3%	189 秒

MC 法は各 Playout 2,000 回

表 2 UCT の正解率
Table 2 Accuracy rate by UCT.

PO	S=20	time	S=30	time	S=40	time
10000	32.2%	293 秒	50.0%	201 秒	62.5%	120 秒
30000	31.1%	1,027 秒	51.5%	739 秒	67.7%	465 秒
50000	30.5%	1,956 秒	51.7%	1,440 秒	69.8%	951 秒

PO：Playout 回数

表 3 UCT+の正解率
Table 3 Accuracy rate by UCT+.

PO	S=20	time	S=30	time	S=40	time
10000	77.6%	345 秒	87.2%	237 秒	88.7%	207 秒
30000	74.8%	1,312 秒	86.7%	943 秒	88.4%	862 秒
50000	74.8%	2,634 秒	86.8%	1,927 秒	88.2%	1,777 秒

PO：Playout 回数

表 4 UCT+対 UCT 対戦結果
Table 4 Result of play between UCT+ and UCT.

黒	白	勝ち	引分	負け
UCT+	UCT	50	0	0
UCT	UCT+	50	0	0
計		100	0	0

数字は UCT+から見た結果

ンダムプレイと大差ない性能であると考えられる．表 2，表 3 を比較すると，提案手法の UCT+は UCT と比較して正解率が大きく向上している．

6.3 対戦実験

UCT+と UCT の先後を入れ替え，それぞれ 50 回計 100 回対戦を行った．1 手の時間は 2 秒で固定している．また，実験時間短縮のため S=45 から完全読み切りを行い勝敗を判断している．結果を**表 4** に示す．勝敗はすべて UCT+から見ての結果である．

このように，Zebra の評価関数を使う UCT+がピュアな UCT を圧倒した．棋譜を見ると，ほとんどの場合中盤で UCT+が有利になり，そのまま押し切っている．

7. 他手法との比較

前章では UCT+が UCT の性能向上に有効な手段であることを示した．本章では提案手法と，他の局面評価関数を用いた探索手法との性能の比較を行う．比較のため，同じ zebra の評価関数を使用する．

7.1 $\alpha\beta$ 探索との比較

基本的なゲーム木探索手法である $\alpha\beta$ 探索をオセロに実装し，UCT+との比較を行う．局面評価関数には zebra を使用するが，局面更新などベース部分は UCT+と同じものを使い，探索などそれ以外の部分はオリジナルで作成した．

7.1.1 正解率による比較

$\alpha\beta$ 探索（評価関数：zebra）の正解率を求めた．結果を**表 5** に示す．

$\alpha\beta$ 探索の正解率は深さ 1 で UCT+を下回るものの，それ以上の深さでは UCT+を上回っている．計算処理時間は深さ 5 でも $\alpha\beta$ 探索の方がかなり短く，UCT+は $\alpha\beta$ 探索の性能には及んでいないことが分かる．

7.1.2 対戦による評価

UCT+との対戦を行い性能を比較した．対戦方法は前章と同じだが，UCT+は 1 手あたりのプレイアウト回数，$\alpha\beta$ 探索は探索の深さを固定して探索を行うものとする．

表 6 に UCT+を用いたプレイヤと $\alpha\beta$ 探索を用いたプレイヤとの対戦結果を示す．これも勝敗の数字はすべて UCT+から見た結果である．

プレーオフ 30,000 回の UCT+は $\alpha\beta$ 探索深さ 1 を圧倒し，深さ 2 には 7 割弱勝つが，深さ 3 には 1 勝もできていない．

表 5 $\alpha\beta$ 探索の正解率
Table 5 Accuracy rate by $\alpha\beta$ search.

深さ	S=20	time	S=30	time	S=40	time
1	77.3%	0.5 秒	84.2%	0.48 秒	82.7%	0.47 秒
2	82.5%	0.55 秒	92.0%	0.55 秒	88.7%	0.52 秒
3	84.8%	1.1 秒	93.5%	1.03 秒	90.3%	0.8 秒
4	88.8%	3.68 秒	94.8%	2.91 秒	92.3%	1.59 秒
5	88.1%	21.6 秒	95.8%	17.1 秒	94.2%	6.7 秒

表 6 UCT+(po=30000) 対 $\alpha\beta$ 探索の勝利数
Table 6 The UCT+(po=30000) vs. $\alpha\beta$ search.

黒	白	勝ち	引分	負け	ノード
UCT+	$\alpha\beta$（深さ 1）	48	1	1	6.94
$\alpha\beta$（深さ 1）	UCT+	42	1	7	10.93
計		90	2	8	
UCT+	$\alpha\beta$（深さ 2）	34	0	16	39.27
$\alpha\beta$（深さ 2）	UCT+	32	0	18	43.14
計		66	0	34	
UCT+	$\alpha\beta$（深さ 3）	0	0	50	281.3
$\alpha\beta$（深さ 3）	UCT+	0	0	50	281.8
計		0	0	100	

数字は UCT+から見た結果
ノード：$\alpha\beta$ 探索の平均探索ノード数

表 7 UCTbreak の正解率
Table 7 Accuracy rate by UCTbreak.

PO	S=20	time	S=30	time	S=40	time
10000	72.3%	290 秒	84.9%	153 秒	78.2%	110 秒
30000	73.6%	929 秒	86.6%	581 秒	74.8%	437 秒
50000	74.7%	2,363 秒	88.1%	1,185 秒	74.2%	913 秒

PO：Playout 回数

7.2 プレイアウト打ち切り＋評価関数との比較

他の評価関数を使う方法として，Lines of Action で使われたプレイアウトの途中で評価関数を呼び評価値が高いプレイヤを勝ちと見なす Lorentz [12] の提案手法（本稿では UCTbreak と呼ぶ）でも実験を行った．評価関数には UCT+と同じ zebra を使用し，プレイアウトを打ち切る深さは葉ノードから 10 手先とする．

7.2.1 正解率による比較

UCTbreak の正解率を求めた．結果を**表 7** に示す．実験結果より，評価関数に zebra を使用した UCTbreak は UCT と比較して正解率が向上したが，UCT+より正解率は低いことが分かった．

7.2.2 対戦による比較

UCT+との対戦を行い性能の比較を行った．対戦方法は前節と同様である．同じ探索時間だと圧倒的な差がついたので UCTbreak の探索時間を長くする実験も行った．

表 8 に UCT+を用いたプレイヤと UCTbreak を用いたプレイヤとの対戦結果を示す．

表 7，表 8 より，提案手法の UCT+は UCTbreak と比

表 8 UCT+(2 秒) 対 UCTZbreak の勝利数
Table 8 The UCT+(2sec) vs. UCTZbreak.

黒	白	勝ち	引分	負け
UCT+	UCTZbreak（2 秒）	50	0	0
UCTZbreak（2 秒）	UCT+	48	1	1
計		98	1	1
UCT+	UCTZbreak（4 秒）	50	0	0
UCTZbreak（4 秒）	UCT+	49	0	1
計		99	0	1
UCT+	UCTZbreak（8 秒）	50	0	0
UCTZbreak（8 秒）	UCT+	50	0	0
計		100	0	0

数字は UCT+から見た結果

較して，圧倒的な性能を示したといえる．

8. 考察

6 章で，まず単純な MC 法より木が成長する UCT 探索が優れていることを確認した．どの手法でも探索開始局面が終局に近いほど正解率が高くなっている．これは終局に近いほど探索されるノードの数が大きく減少し，よりゲーム終了時の結果が反映されるためであると思われる．S=40 ではほとんど打てる場所は限られているのでピュアな UCT でも高い正解率になっている．MC 法と UCT の S=20, 30 ではランダムプレイとほとんど変わらない結果となってしまっている．これは S=20 や 30 の末端ノード数は S=40 と比べて非常に多く勝率だけで手を選択するにはプレイアウト数が足りないためであると思われる．なお，S=30 から S=40 まで，それぞれの局面で打てる場所が 10 カ所以上あることが多く，S=30 の探索木ノード数は S=40 と比べると 10^{10} 以上は多いと考えられる．

UCT+は勝率だけの UCB 値を用いる UCT に比べて正解率が大きく向上している．特に S=20 が最も正解率が向上している．また，対戦実験では UCT+が UCT を圧倒した．このことから勝率だけでなく，局面評価関数を用いて手を選択することはオセロにおいては非常に有効な手段であることが分かる．局面評価関数を UCB 値にどのように足すべきかは難しく，本稿では式 (3) のように試行回数の平方根で割り算をしたが，実用的にどのような方法が良いかはゲームの種類や制限時間などによってそれぞれ検証すべきだと思われる．

また，7 章では，提案手法とその他の手法の性能を比較し，UCT+の位置づけを示した．実験の結果，本稿の UCT+（プレーオフ 30,000 回）は局面評価関数に zebra を使用する $\alpha\beta$ 探索の深さ 2 より強く，それ以上の深さの $\alpha\beta$ 探索にはまったくかなわないことが分かった．$\alpha\beta$ 探索の探索ノード数は深さ 3 でも数百のオーダで，やはりオセロではモンテカルロをベースとした手法よりも単純なミニマックス探索が向いているのではないかと思われる．だが，同じ UCT で局面評価関数を使う手法である UCTbreak よりは UCB 値に直接局面評価関数を加える UCT+の方がオセロでは優れた性能を持つことが分かった．Lines of Action のようにプレイアウトがいつ終わるかわからないゲームでなければ，UCT+の方が局面評価関数の使い方としては有効であると予想できる．

9. 今後の課題

提案手法がオセロにおいて有効なことを示せた．今後は優れた評価関数の存在する他のゲームでも実験を行い，どの程度有効であるか調べてみたい．特に近年モンテカルロ将棋の研究がさかんになってきているので，将棋でどの程度有効か試したい．トランプなど不完全情報多人数ゲームでも試してみたい．

また，今回 E_i をプレイアウトが進むごとに小さくなるよう $\sqrt{n_i}$ で割ったが，どのように局面評価関数を UCB 値に加えるのが良いのか，オセロや他のゲームでさらに検証を行い，実用的にどのような手法が良いか検討したい．

参考文献

[1] Buro, M.: How Machines have Learned to Play Othello, *IEEE Intelligent Systems Journal*, Vol.14, No.6, pp.12–14 (1999).

[2] Buro, M.: The Othello Match of the Year: Takeshi Murakami vs. Logistello, *International Computer Chess Association* (*ICCA*) *Journal*, Vol.20, No.3, pp.189–193 (1997).

[3] Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search, *Proc. 5th International Conference on Computers and Games*, Lecture Notes in Computer Science (LNCS), Vol.4630, pp.72–83 (2006).

[4] Kocsis, L. and Szepesvari, C.: Bandit based Monte-Carlo Planning, *Proc. 17th European Conference on Machine Learning* (*ECML 2006*), pp.282–293 (2006).

[5] Auer, P., Cesa-Bianchi, N. and Fischer, P.: Finite time Analysis of the Multi-armed Bandit Problem, *Machine Learning*, Vol.47, pp.235–256 (2002).

[6] 橋本隼一，橋本 剛，長嶋 淳：コンピュータ将棋におけるモンテカルロ法の可能性，第 11 回ゲームプログラミングワークショップ，pp.195–198 (2006).

[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 (2008).

[8] 竹内聖悟，金子知適，山口和紀：将棋における，評価関数を用いたモンテカルロ木探索，第 15 回ゲームプログラミングワークショップ，pp.86–89 (2010).

[9] Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: Modifications of UCT with Patterns in Monte-Carlo Go, Technical Report 6062, pp.1–19, INRIA (2006).

[10] Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, *International Computer Game Association* (*ICGA*) *Journal*, Vol.30, No.4, pp.198–208 (2007).

[11] 松井利樹，野口陽来，土井佑紀，橋本 剛：囲碁における勾配法を用いた確率関数の学習，情報処理学会論文誌，Vol.51, No.11, pp.2031–2039 (2010).

[12] Lorentz, R.: Amazons Discover Monte Carlo, *Proc. 6th International Conference on Computers and Games*,

Lecture Notes in Computer Science (LNCS), Vol.5131, pp.13–24 (2008).
[13] Kloetzer, J., Iida, H. and Bouzy, B.: Playing Amazons Endgames, *International Computer Game Association* (*ICGA*) *Journal*, Vol.32, No.3, pp.140–148 (2009).
[14] Winands, M.H.M. and Bjornsson, Y.: Evaluation Function Based Monte-Carlo LOA, *Advances in Computer Games* (*ACG 2009*), Lecture Notes in Computer Science (LNCS), Vol.6048, pp.33–44 (2009).
[15] Tristan, C.: Nested Monte-Carlo Search, *Proc. 21st International Joint Conference on Artificial Intelligence* (*IJCAI-09*), pp.456–461 (2009).
[16] 秋山晴彦，小谷善行：Nested Monte-Carlo 探索の AMAF を用いた探索数調整による改良，情報処理学会ゲーム情報学研究会報告，Vol.2010, No.7, 2009-GI-23, pp.1–7 (2010).
[17] Andersson, G.: Gunnar's Othello page, available from ⟨http://radagast.se/othello/⟩.

推薦文

情報処理学会中国支部表彰規定に則り，平成 23 年度（第 62 回）電気・情報関連学会中国支部連合大会で発表された中から特に優秀であることが認められた優秀論文発表賞を受賞した論文である．（中国支部長　三池秀敏）

橋本 剛　（正会員）

1970 年生．1993 年京都大学農学部卒業．1996 年東京大学大学院理学系研究科修士課程在学中に中国雲南民族学院へ留学．1997 年同大学院中途退学．2002 年静岡大学大学院理工学研究科博士後期課程修了．博士（工学）．同年学術振興会特別研究員 PD．2003 年カナダアルバータ大学客員研究員．2005 年北陸先端科学技術大学院大学情報科学研究科講師．2010 年松江工業高等専門学校情報工学科准教授．コンピュータ将棋等，ゲーム情報学研究に従事．ICGA，コンピュータ将棋協会，ロボカップ日本委員会各会員．

前原 彰太

1987 年生．2009 年島根大学総合理工学部数理・情報システム学科卒業．2011 年同大学大学院総合理工学研究科数理・情報システム学専攻修士課程修了．同年株式会社アイティフォー入社．

川島 哲哉

2010 年島根大学総合理工学部数理情報システム学科卒業．2012 年島根大学大学院総合理工学研究科数理情報システム学専攻修了．同年株式会社リゾーム入社．

小林 康幸　（正会員）

1971 年大阪市立大学理学部数学科卒業．1973 年大阪市立大学大学院理学研究科修士課程修了．理学博士．1974 年広島大学理学部勤務．現在，島根大学総合理工学研究科情報システム学領域教授．ゲーム理論，計算機統計学の研究に従事．計算機統計学会，応用統計学会，日本統計学会各会員．