Vol.2010-GI-24 No.5
2010/6/25

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

前原彰太[†1] 橋本剛[†2] 小林康幸[†1]

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，局面評価関数を使ったUCTの研究はこれまでなかった．

本研究では新たに局面評価関数をUCB値に加える手法を提案する．実験では比較的簡単に局面評価関数が作れるオセロに提案手法を実装し評価を行った．その結果，提案手法は圧倒的な性能を示しその有効性が実証された．

A new UCT search method using position evaluation function and its evaluation by Othello

Shota Maehara,[†1] Tsuyoshi Hashimoto[†2] and Yasuyuki Kobayashi[†1]

The Monte Carlo tree search, particularly UCT, gains a great success and is being widely studied as a game tree search method. However UCT using position evaluation function has been studied because of the difficulty of calculating position evaluation function in the game of GO, the main target of UCT research.

We propose a new method that adds position evaluation function to the UCB value in this paper. It is implemented for the game of Othello that is relatively easy to make position evaluation function and experiments are performed. The results show the overwhelming ability of proposed method and its effectiveness is verified.

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もパターンの学習による評価関数を用いているが，探索手法はミニマックス法を基本としたものである．パターンの学習を用いたプログラムはすでに人間では勝てない強さ[⋆1]ではあるが，まだオセロは解明されていない．近年，コンピュータ囲碁ではモンテカルロ木探索[1]が注目を集めている．モンテカルロ法と呼ばれる乱数を用いたプレイヤー同士で終局までゲームを行ない，その結果を局面の評価と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[2]がある．UCTはUCB[3]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．

モンテカルロ法は囲碁やオセロのようにプレイヤーがお互いにランダムに手を選択しても終局を迎えることができるゲームで適用しやすい手法である．いつ終わるかわからずランダムだと収束しにくい将棋ではモンテカルロ法の使用は難しいことが知られている[4][5]．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有効な手法として注目され，研究が盛んに行なわれている[1][2][6]．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ー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）を実現する事で強化を図ることが盛んに行われている．一方，囲碁以外のゲームでUCTを使った研究も現在は盛んで，Amazons[7][8]やLOA[9]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[10][11]など多岐に渡っている．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

†1 島根大学大学院総合理工学研究科
Graduate School of Science and Engineering, University of Shimane
†2 松江工業高等専門学校情報工学科
Department of Information Engineering, College of Technology of Matsue

⋆1 M.Buroの Logistelloが1997年に当時の世界チャンピオンを破る

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．コンピュータオセロはかなりモンテカルロ法は適した題材であるが，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．

2. UCB 値とモンテカルロ木探索

モンテカルロ法は全ての手を乱数によって選択していく．しかし，ゲームにおいて明らかに悪い手は結局選ばれないので，できるだけ探索しないほうがよい．計算量を小さく保ちつつ，良さそうな手だけに探索を割り当てる方法として UCB（Upper Confidence Bound）が考案された[3]．UCB 値が最も高いノードを選び続けることで，良さそうなノードにより多くの探索が割り当てられる．

UCB 値は，ある局面で i 番目の手に n_i 回のプレイアウトが行なわれた時の勝率を $\overline{X}_i$，n をある局面で行なわれたプレイアウト全ての回数，c をゲームごとに調整を行なう何らかの係数としたとき，以下のように表される．

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

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[3] だが，本稿では式 (1) を UCB 値として進めていく．

モンテカルロ木探索とは，ある一定数の探索回数を超えたノードは展開され，その子ノードが記憶され，そこからプレイアウトが行なわれる探索法である．

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)[2] である．

UCT 探索では UCB 値の高いノードが多く選ばれ，ある閾値を超えるとその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事が多い．最近は評価値の機械学習による計算が主流で，囲碁を中心に盛んに研究されている[12][13]．Amazons でも同様の研究が行われて成果を上げている[8]．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事も行われている[13]．碁では局面評価関数の作成が難しいため古典的な MINMAX からモンテカルロ法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 7) では，MINMAX 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すというもので，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[4][5]．

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終局の勝ちか負けの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 値とは別に用いた手法が多く考案されてきている．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．

本研究では 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，これを UCT+と名付ける．

任意の局面で i 番目の手に n_i 回プレイアウトが行なわれたとき，i 番目の手の評価値を E_i とする．E_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．

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

UCT+は，式 (2) を最大化するノードを選択する．UCT+は式 (1) に直接，局面評価関数を加えるという点で，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なった．オセロは必ず一定の手数で収束し，ピュアな UCT で間単に強いプログラムを作る事ができる．またすでに多くの優れた局面評価関数が知られており，本研究の題材に適

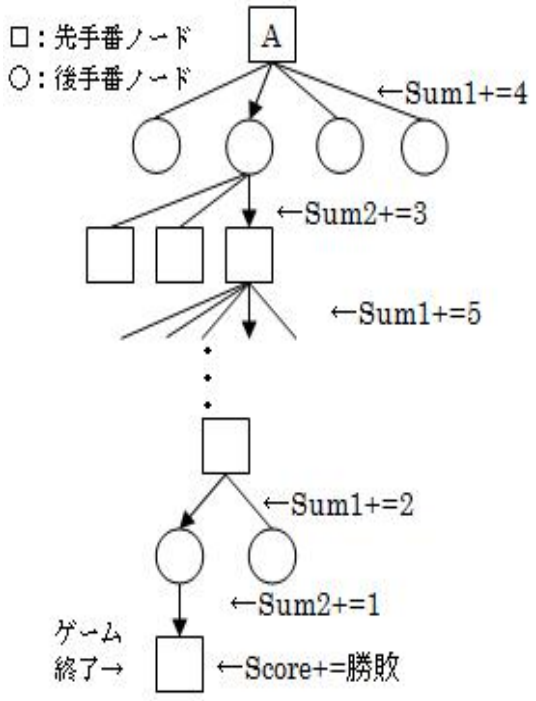

図 1 提案手法におけるプレイアウト

していると思われる．

本稿ではプレイアウトと同時に合法手の数をカウントすることで簡単に計算できる大浦の手法[15] を使う．オセロでは合法手の多いプレイヤーが有利であるということが知られている（詳しくは文献 14) などのオセロ入門書を参照）．大浦は終局の情報に加えてプレイアウト中の合法手数を用いた評価関数を用いることで良い結果を得ることに成功している．本稿では同文献より変数名を一部そのまま引用している．以下その手法を詳しく説明する．

$Sum1$ は先手の子ノード数，$Sum2$ は後手の子ノード数，$Score$ は終局での勝ち (+1) か負け (−1) か引き分け (0) を表す変数とする．

合法手の計測方法は図 1 のようになる．局面 A からプレイアウトが行なわれたとする．このとき，乱数を用いてゲームを進めるときにそのノードが着手できる数を計測しておく．1 回のプレイアウトが終わったときに，$Sum1$ には局面 A からの先手の子ノードの総数，$Sum2$ には後手の子ノードの総数，$Score$ には勝ちか負けかによる値が記憶される．数回のプレイアウトが終わった後に $Sum1$，$Sum2$，$Score$ それぞれの平均値を計算する．$Score$ が式 (2) の $\overline{X}_i$，$Sum1$ と $Sum2$ が E_i に相当する．

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 $Sum1$ と $Sum2$ 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．本稿では判別分析を用いて実験的に適当な係数を算出した．

なお判別分析とは，2 つ以上のグループの間にある違いを見分けるための手法である．ここでは $Score$，$Sum1$，$Sum2$ という 3 組の変量を用いて勝ちと負けの 2 つのグループを見分ける．判別分析によって 3 変量それぞれに適当な係数が与えられ，それが判別関数と呼ばれる上手くグループ分けを行なう関数になる．

本稿では判別分析によって i 番目のノードで得られる評価値である $Sum1_i$，$Sum2_i$，$Score_i$ の係数を算出し，その比率を基準にして UCT+で用いる局面評価関数を次の式 (3) のように定めた．

$$E = \frac{Sum1_i - Sum2_i}{30}, c = \frac{8}{3}$$

$$(\overline{X}_i + \frac{Sum1_i - Sum2_i}{30}) + \frac{8}{3}\sqrt{\frac{2\log n}{n_i}} \quad (3)$$

式 (2) の E_i と c は扱うゲームと，用いる変数によって適当な値をつけなければ UCT+は機能しない．

6. 実　　験

6.1 実験方法

予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 300 件を用いる．これはオセロの棋譜から，調べたい探索開始局面において必ず完全探索の値が勝ちであるノードと負けとなるノードが 1 つずつは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ーである scrzebra[⋆1]によって行なった．このプログラムはミニマックス法に枝刈りを加えた，α β法を基本としたアルゴリズムを用いて完全探索を行なっている．

S は探索開始局面を表すこととする．S=40 と書いている場合，初手から 40 手目が終了した局面から探索を開始しているということである．

例として，S=50 は以下のように書き込まれている．

(g,1),-10, (h,1),14, (h,2),-10, (h,3),0 (h,7),2,

括弧の中が着手できる手の場所，値が終局の石の差（先手－後手）である．

図 2 は上記の例の実際のオセロの盤面である．この例では先手は黒であり，着手できる場所は 5 か所ある．ファイルにはその着手できる場所にお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 (h,1) か (h,7) を打った後，お互いに最善手を打てば先手が

⋆1 http://radagast.se/othello/download2.html

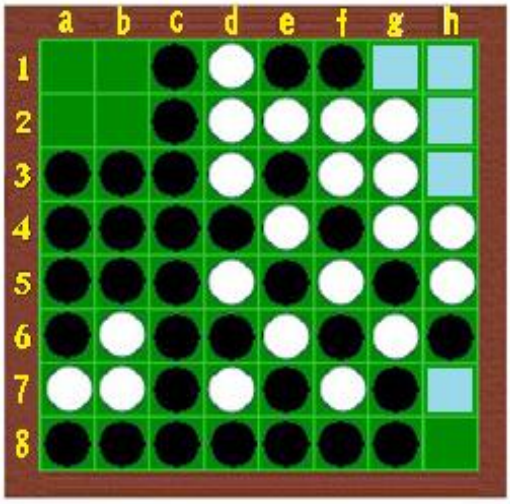

図 2 棋譜ファイルの例

表 1 UCT による正解率 (単位：%)
Table 1 Right answer percentage by UCT.

	S=30	S=40
Playout=10000	47.83	56.33
Playout=30000	50.17	63.50

表 2 UCT+による正解率 (単位：%)
Table 2 Right answer percentage by UCT+.

	S=30	S=40
Playout=10000	82.17	82.50
Playout=30000	87.5	88.33

勝てることになる．

選んだ手の，完全探索結果の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，以下の環境で実験を行なった．

実験環境

- OS Debian Linux5.0.3
- CPU Pentium4 3.2GHz
- メモリ 512MB
- 言語 C

6.2 実験結果

それぞれの探索手法の実験結果を表 1，表 2 に載せる．なお，表 1 の単純な UCB 値のみを用いた UCT では UCT+と比較するために c は UCT+と同じ値にしている．

比較のために別の手法による正解率を表 3 に載せる．

表 3 単純な手法の正解率 (単位：%)
Table 3 Right answer percentage by simple method.

	S=30	S=40
ランダムプレイ	45.30	46.70
モンテカルロ法 (Playout=2000)	45.00	57.33

ランダムプレイは合法手の中から完全にランダムに 1 手選ばせている．これにより，正解率測定に用いている棋譜ファイルの合法手にどの程度偏りがあるのかが分かる．結果としては少しだけ先手に不利であるが，正解率が 50% に近いので手法ごとの正解率の比較を行なうにあたり，使用する棋譜は公平性をもっていることが分かる．モンテカルロ法は木が成長しないので探索数を多くしても正解率はほとんど変わらない．

7. 考　　察

表 1，表 2 ともに探索数が多いと正解率も高くなっている点は UCT 探索の特長といえるがオセロにおいても確認できた．また，表 1 と表 3 より木が成長しないモンテカルロ法より木が成長する UCT 探索が優れていることも確認できた．そして，UCT+は勝率だけの UCB 値を用いている UCT に比べて正解率が大きく向上しているので，オセロにおいては有効であることが分かる．

提案手法はオセロだけでなく，Amazons・チェス・将棋・囲碁や 2 人ゲームでないゲームでも，有効な局面評価関数を見つけ上手く UCB 値に加えることで，良い結果が得られるのではないかと考えられる．

8. 今後の課題

提案手法は，良い局面評価関数を UCB 値に加えるだけなので，局面評価関数の精度が良ければ，そのまま強さに反映されると思われる．そのため，オセロに関しては Logistello のような強いオセロプログラムの評価関数を用いることができれば，より強くなると予想されるので，実装してみたい．

また，考察で記したように他のゲームでも有効な手法だと思われるので，こちらも実装していきたい．

参 考 文 献

1) Coulom, R.: Efficient selectivity and backup operators in monte-carlo tree search, *Proceedings of the 5th International Conference on Computers and Games*, Turin, Italy (2006).
2) Kocsis, L. and Szepesvari, C.: Bandit based Monte-Carlo Planning, *Proceedings of the 15th European Conference on Machine Learning*, pp.282–293 (2006).
3) 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).
4) 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, Proceedings of The 11th Game Programming Workshop pp. 195-198 (2006).
5) 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).
6) Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: Modifications of UCT with Patterns in Monte-Carlo Go, Technical Report RR-6062, INRIA (2006).
7) Lorentz, R.: Amazons Discover Monte Carlo, Computers and Games, Lecture Notes in Computer Science, Vol. 5131, pp.13–24, (2008).
8) Julien Kloetzer, Hiroyuki Iida and Bruno Bouzy: Playing Amazons Endgames, ICGA Journal, To be appear,
9) Winands, M.H.M. and Bjornsson, Y. (2010): Evaluation Function Based Monte-Carlo LOA, In Advances in Computer Games (ACG 2009), Lecture Notes in Computer Science (LNCS 6048), pp.33–44. c Springer, Berlin Heidelberg.
10) Tristan Cazenave: Nested Monte-Carlo Search, IJCAI2009, pp.456-461(2009).
11) 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).
12) Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go, In Computer Game Workshop, Amsterdam, The Netherlands (2007).
13) 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, 情報処理学会ゲーム情報学研究会報告, Vol.2009, No.27, 2009-GI-21, pp.33–40 (2009).
14) 谷田邦彦: 図解 早わかりオセロ, 日東書院.
15) 大浦教彰: 標本抽出法の改良について, 島根大学平成 17 年度卒業論文 (2005).