Vol.2020-GI-43 No.24
2020/3/14

モンテカルロ木探索における評価関数へのノイズ付加による影響

佐原 和就[1,a)]　松崎 公紀[1,b)]

概要：モンテカルロ木探索は囲碁やチェスをはじめとしたコンピュータプレイヤの強さに大きく貢献した．モンテカルロ木探索を効果的にする手法として評価関数の併用が挙げられる．しかし，完璧な評価関数を作ることは現実的には不可能であり，評価値の正確さがモンテカルロ木探索の正確さに何かしら影響するはずである．本研究ではオープンソースのオセロプログラム「Zebra」の評価関数を用いてオセロを対象にこの問題に取り組む．Zebra の評価関数より算出された評価値に対しノイズを付加することで，不正確な評価関数を用いるモンテカルロ木探索プレイヤのレーティングを計算し，評価値の正確さが及ぼす影響を検証する．検証結果，ノイズを付加することでレートが減少する傾向が確認され，ノイズの大きさがより大きくなればレートも減少することが確認された．また，シミュレーション数を増やすことで若干ではあるが，レートが向上することがわかった．

キーワード：オセロ，モンテカルロ木探索，評価値，ノイズ

1. はじめに

モンテカルロ木探索 [9] は，2006 年に R．Coulom が囲碁プログラム「Crazy Stone」に用いるために提案した探索アルゴリズムであり [7]，「Crazy Stone」は同年 8 月の Computer Olympiad の囲碁 9 路盤部門で優勝した．その後，L. Kocsis と C. Szepesvári [8] によって UCT (UCB1 applied to Trees) アルゴリズムが提案され [1]，さまざまなゲームに対してモンテカルロ木探索が適用されてきた．モンテカルロ木探索では，ある局面から仮想的に終局まで着手する「プレイアウト」を繰り返し行う．ランダムなプレイアウトを用いる UCT アルゴリズムでは，プレイアウトを無限回繰り返すことで最適解に収束することが証明されている．

モンテカルロ木探索による多くのプレイヤでは，限られた時間内で効率良く適切な手を決定するため，各局面に対する評価関数を用いて木探索やプレイアウトを制御する方法をとっている [6], [10], [11]．ランダム性が制限されたこれらの評価関数を用いる手法では，必ずしも最適解への収束は保証されないものの，多くの場合において優れたプレイヤが得られている．たとえば，DeepMind 社による AlphaGo [12] / AlphaGo Zero [13] では，ニューラルネットワークによって設計された方策（policy）関数と値（value）関数を併用することにより，人間を超える囲碁プレイヤの開発に成功している．

しかしながら，完全に正確な評価関数を得ることは現実的に不可能である．したがって，用いる評価関数に含まれる誤差がモンテカルロ木探索プレイヤの強さに何らかの影響を及ぼすと考えるのは自然であるが，著者が知る限りこの課題に取り組んだ研究はほとんどない．松崎と北村は，オープンソースのオセロプログラム「Zebra」[3] の優れた評価関数の内部動作を制限することで，10 通りの（特異的な誤差を含む）評価関数を設計して，この課題に取り組んだ．そのうち，文献 [2], [14] では node prior アルゴリズムや early playout termination アルゴリズムにこれらの評価関数を用いた場合について，文献 [4] では PUCT アルゴリズムに用いた場合について，それぞれ評価を行っている．

本研究では，文献 [4] と同様に，誤差を含む評価関数を PUCT アルゴリズムに用いた場合のプレイヤの強さについて評価を行う．文献 [4] における評価関数との違いは，正規分布により発生させた誤差を与えることである．具体的には，局面によらず一定の大きさの誤差を与える場合と，終局に近い局面により小さな誤差となるよう制御した誤差を与える場合の 2 通りについて詳細に実験を行う．

これらの実験の結果得られた知見は大きく以下の 3 つである．

[1] 高知工科大学
[a)] 200319x@ugs.kochi-tech.ac.jp
[b)] matsuzaki.kiminori@kochi-tech.ac.jp

- 誤差を含む評価関数を用いた場合，含まれる誤差の大きさに対しておよそ線形にプレイヤの強さが低下する．
- α-β アルゴリズムと比較して，PUCT アルゴリズムでま評価関数に含まれる誤差の影響が大きい．
- 誤差を含む評価関数を用いた場合，PUCT アルゴリズムではシミュレーション数を増やしてもプレイヤの強さがあまり向上しないことがある．

2. 要素技術

2.1 モンテカルロ木探索

本研究では，モンテカルロ木探索のアルゴリズムとして，AlphaGo Zero [13] で用いられた PUCT アルゴリズムを用いる．

AlphaGo Zero で用いられた PUCT アルゴリズム（以降，単に PUCT アルゴリズムと呼ぶ）では，方策（policy）関数と値（value）関数の 2 つの評価関数を用いる．探索木の各枝 (s,a) では平均行動値 $Q(s,a)$，訪問回数 $N(s,a)$，および事前確率 $P(s,a)$ の 3 つの値を保持する．PUCT アルゴリズムの各シミュレーションでは，以下のステップで計算を行う．

(1) 選択：根ノードから葉ノードへとたどる．ここで各ノードでは，平均行動値と以下で説明するバイアス値の和が最大となるものを選択する．
(2) 展開：選択された葉ノードの訪問回数が閾値を超えていれば，その葉ノードを展開して，子ノードを追加する．本研究の実装では閾値を 1，すなわち，2 度目の訪問で展開し，その子ノードのひとつを選ぶ．
(3) 評価：選択された葉ノード s の局面評価値 $V(s)$ をシミュレーションの結果とする[*1]．
(4) 更新：根ノードから葉ノードまでの経路上のノードについて，平均行動値 $Q(s,a)$，訪問回数 $N(s,a)$ を更新する．

PUCT アルゴリズムでは，選択ステップにおいて以下の式が示すように，平均行動値とバイアス値の和が最大のものを選択する．

$$a' = \underset{a}{\operatorname{argmax}} \left(Q(s,a) + c \frac{P(s,a)}{1+N(s,a)} \right)$$

ここで定数 c は，本研究では $c = 19.5$ とした．

各ノードの局面評価値 $V(s)$ と各枝の事前確率 $P(s,a)$ は，ノードが展開されるときにそれぞれ一度だけ評価される．局面評価値 $V(s)$ には，次節の評価関数の結果をそのまま用いる．一方，事前確率 (s,a) は，評価関数の結果を 0.0512 倍した上で softmax により確率に変換して用いる．

[*1] 多くのモンテカルロ木探索のようにプレイアウトを行わないため，本研究の PUCT アルゴリズムは評価関数が決定的であれば決定的に動作する．

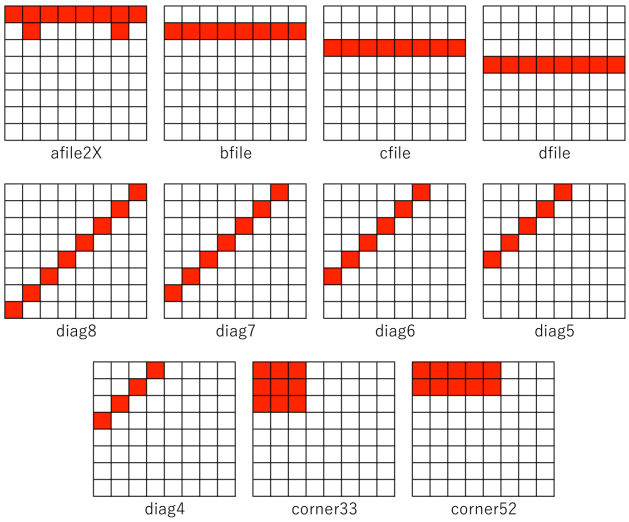

図 1 Zebra の評価関数の抽出パターン

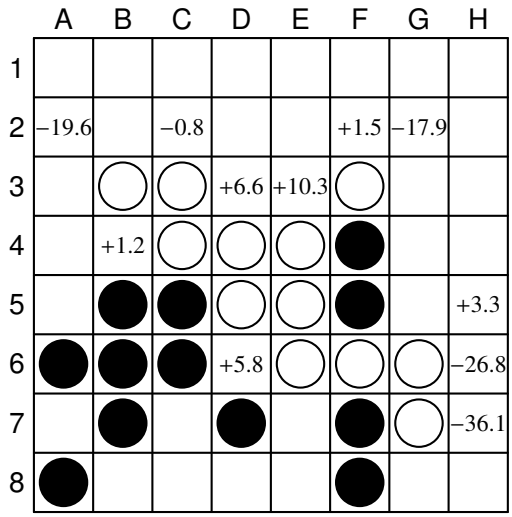

図 2 Zebra の評価値の例：次黒番の評価値を示している

2.2 Zebra の評価関数

「Zebra」[3] はオープンソースのオセロプログラムであり，チャンピオンレベルの強さをもつものである．Zebra では，比較的高速かつ正確な評価関数を用いて，α-β 法により着手を判断している．Zebra で用いられる評価関数は，盤面から局所的な特徴を抽出し，それらの評価値の和として与えられる．各局所的特徴に対する評価値は，多数の対戦実験の結果から機械学習することで得られたものである．**図 1** に特徴を抽出するための 11 のパターンを示す．図において，色のついたマスの状態によって局所的特徴が決定される [5]．

Zebra の評価関数では，1 コマ差を 512 とする整数値を用いている[*2]．本論文では，評価値の値はオセロのコマ数でスケーリングした小数を用いることとする．**図 2** に，評価関数を適用した結果について例を示す．

[*2] 本研究では，実装の都合上，同等のスケールでの浮動小数点数を用いている．

3. 実験方法

本研究では，Zebra の評価値に正規分布に従う誤差を付加したものを返す評価関数を 2 通り実装し，正規分布の分散を変えることで，元の Zebra の評価関数を含め 41 通りの評価関数を実現した．次に，これらの評価関数を用いる α-β プレイヤと PUCT プレイヤを実装した．プレイヤのパラメータの違いにより，α-β プレイヤ 5×41 通り，PUCT プレイヤ 8×41 通りの計 533 プレイヤを作成した．これらのプレイヤ間で先手・後手を交換してリーグ戦（283,556 ゲーム）を行い，その結果から Elo レーティングにより各プレイヤの強さをレート値として求める．

3.1 ノイズを含む評価関数

本研究では，Zebra の評価値に正規分布に従う誤差を付加したものを返すことで，ノイズを含む評価関数を与えた．ノイズの標準偏差の時間変化の観点で 2 通りの実装方法を考える．

減衰なし ゲームの序盤・中盤・終盤によらず，一律の標準偏差 σ の正規分布に従うノイズを付加する．

減衰あり 多くのゲームでは一般に終盤に近づくほどより正確な評価関数を得ることが容易である．したがって，ゲームの進行度（手数）に対して線形に減少し，60 手目で 0 となるようなノイズを付加する．具体的には，評価対象の局面が k 手目であるとき，$\sigma' = \sigma \times (60-k)/60)$ と補正した標準偏差を用いる．

ノイズの標準偏差が減衰しない評価関数では，標準偏差の値として 0.195 から 3.91 まで 0.195 ずつ変えて用いた．ノイズの標準偏差が減衰する評価関数では，標準偏差の値として 0.488 から 9.77 まで 0.488 ずつ変えて用いた．

ノイズを付加した評価値は，同一ゲーム中で出現した同一盤面では同じ値が返るようにキャッシュすることとした．これは，一般にプレイヤが用いる評価関数はひとつであり，その評価値は入力盤面に対して静的に決まるのが適切であると考えたからである．

3.2 プレイヤ

実験に用いるプレイヤには，α-β アルゴリズムによるものと PUCT アルゴリズムによるものを用いる．

α-β プレイヤ 先読みの深さ d を，2, 3, 4, 5, 6 の 5 通りとする．

PUCT プレイヤ シミュレーション数 n を，50，100，200，500，1000，2000，5000，10000 の 8 通りとする．

4. 実験結果と考察

α-β プレイヤの実験結果を**図 3** と**図 4**，PUCT プレイヤの実験結果を**図 5** と**図 6**，シミュレーション数による実験結果を**図 7** と**図 8** に示す．

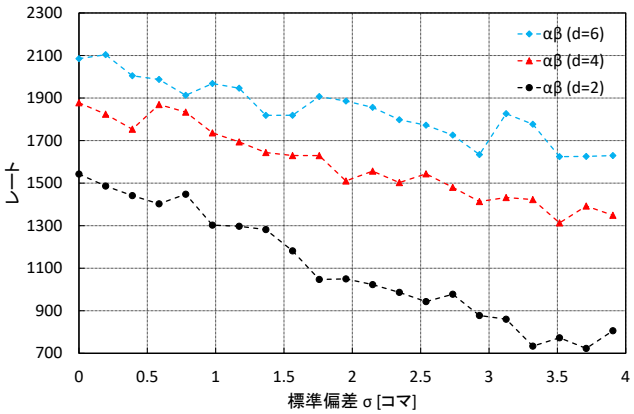

図 3 減衰なしのノイズを付加した評価関数を用いた α-β プレイヤのレート

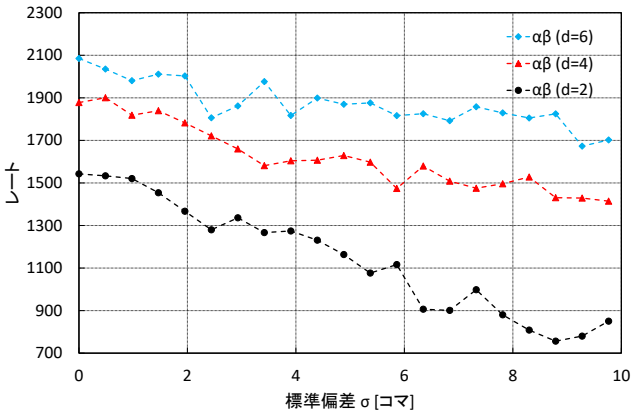

図 4 減衰ありのノイズを付加した評価関数を用いた α-β プレイヤのレート

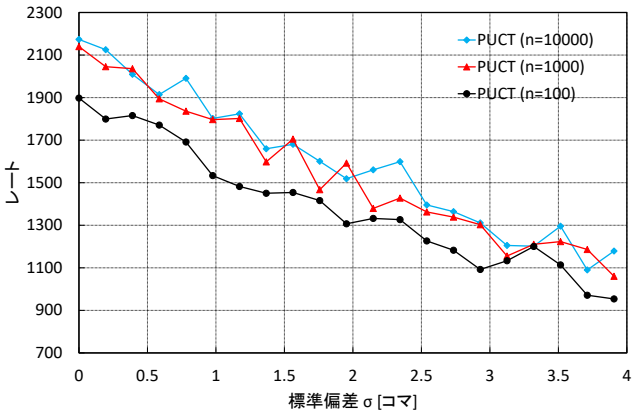

図 5 減衰なしのノイズを付加した評価関数を用いた PUCT プレイヤのレート

α-β アルゴリズムと PUCT アルゴリズムの両方で，評価関数に付加するノイズが大きくなるとレートが小さくなることが確認できた．とくに，いずれのプレイヤでも，実験で調査した範囲のノイズの大きさでは，ノイズの大きさとレートの関係がおよそ線形という結果になった．一方でその（負の）傾きについては興味深い違いがある．

α-β アルゴリズムでは，探索深さ d を大きくすると，傾きの絶対値が小さくなる．（減衰なしの場合 -217（$d = 2$）

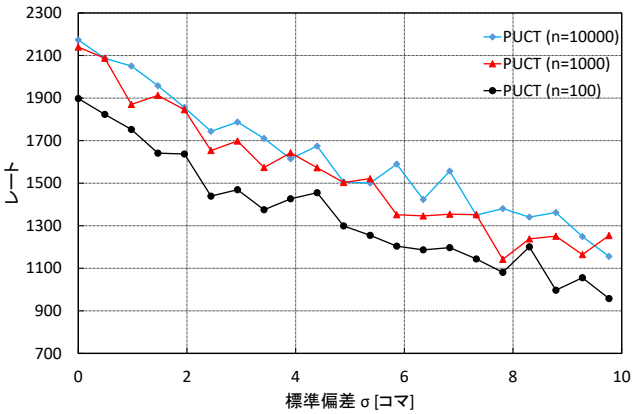

図 6 減衰ありのノイズを付加した評価関数を用いた PUCT プレイヤのレート

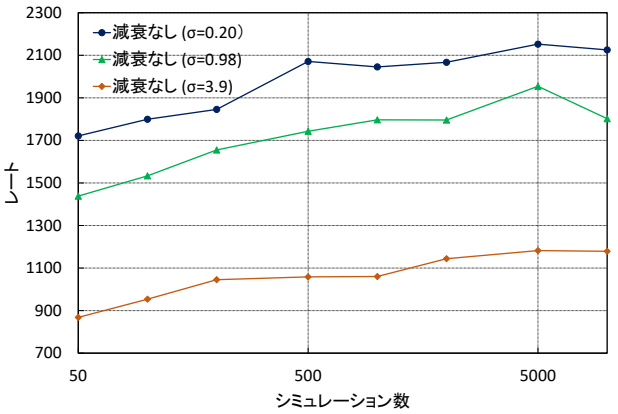

図 7 シミュレーション数に対する PUCT プレイヤのレート（減衰なしのノイズを付加した場合）

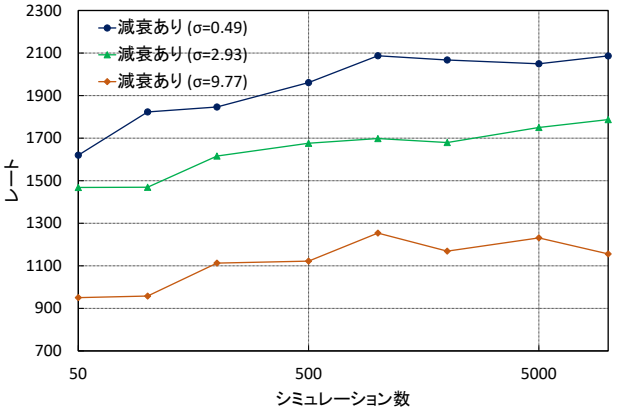

図 8 シミュレーション数に対する PUCT プレイヤのレート（減衰ありのノイズを付加した場合）

$\Rightarrow -111$ $(d=6)$ ／減衰ありの場合 -85.0 $(d=2)$ $\Rightarrow -30.1$ $(d=6)$）

一方，PUCT アルゴリズムでは，シミュレーション数を増やすと，ほぼ変わらないかわずかに大きくなる結果となった．（減衰なしの場合 -228（$n=100$） $\Rightarrow -266$（$n=10000$）／減衰ありの場合 -86.6（$n=100$） $\Rightarrow -91.4$（$n=10000$））

α-β アルゴリズムでは，探索深さ d を大きくすると性能向上が見られる．一方，PUCT では，シミュレーション数 $n=100$ から $n=1000$ では性能向上があるものの，$n=1000$ から $n=10000$ ではほとんど変化がない．このことは，シミュレーション数とレートの関係を示すグラフからより明確に分かる．各条件において，シミュレーション数 500 から 1000 より先で性能向上が見られない．この結果から，PUCT において計算時間を増やすときには，シミュレーション回数を増やすよりもそれぞれの評価値の値をより正確に計算することが重要である．

5. 関連研究

モンテカルロ木探索を用いるプレイヤでは，限られた時間内で効率良く適切な手を決定するため，各局面に対する評価関数を用いて木探索やプレイアウトを制御する方法をとっている．ランダム性が制限されたこれらの評価関数を用いる手法では，必ずしも最適解への収束は保証されないものの，多くの場合において優れたプレイヤが得られている．しかしながら，完全に正確な評価関数を得ることは現実的に不可能である．したがって，用いる評価関数に含まれる誤差がモンテカルロ木探索プレイヤの強さに何らかの影響を及ぼすと考えるのは自然であるが，著者が知る限りこの課題に取り組んだ研究はほとんどない．松崎と北村の研究 [2] では，Zebra の評価関数の評価パターンをいくつかを除去した評価関数で不正確な評価関数をプレイアウト中の着手と UCB1 値に適用した場合の影響を検証している．評価パターンを除去することで偏りのある誤差での検証を行っており，結果として従来の Zebra の評価関数が検証範囲では最も優れており，パターン除去した評価関数はあまり影響が出ない場合や大きく勝率が向上している結果となっており，UCB1 値への評価関数の適用方法が良いものではなかったとされている．

本研究では，α-β アルゴリズムと評価関数の正確さが大きく表れる PUCT アルゴリズムを用いて不正確な評価関数の影響について検証した．また，評価値に対してノイズを付加することで偏りのない誤差で検証を行っている．

6. まとめ

誤差を含む評価関数を用いることで，含まれる誤差の大きさに対しておよそ線形にプレイヤの強さが低下する．また，誤差による影響は α-β アルゴリズムより PUCT アルゴリズムに大きな影響が及ぶ．さらに，誤差を含む評価関数を用いた場合，PUCT アルゴリズムではシミュレーション回数を増やしても必ずプレイヤの強さが大きく改善するとは限らない．

今後の課題として，引き続きノイズの大きさを増やすことで強さの減少が一定のところで収まるのか，PUCT アルゴリズムより評価関数の影響を受けにくいプレイヤでの場合，評価関数自体の変更などを行い，評価関数の正確さが

及ぼす影響力についてより正確な原因を究明できると考えられる．

謝辞 本稿の実験は，高知工科大学の IACP クラスタによって実施されたものである．

参考文献

[1] 松原 仁: 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.
[2] 北村 直輝, 松崎 公紀: 評価関数の違いがモンテカルロ木探索プレイヤの強さに与える影響. 第 59 回プログラミング・シンポジウム予稿集, pp 173–183, 2018.
[3] Gunnar Andersson: Zebra, 入手先 ⟨http://radagast.se/othello/index.html.⟩
[4] Kiminori Matsuzaki: Empirical Analysis of PUCT Algorithm with Evaluation Functions of Different Quality, Conference on Technologies and Applications of Artificial Intelligence, 142–147 2018
[5] Gunnar Andersson, 訳 奥原 俊彦: 強いオセロプログラムの内部動作, 入手先 ⟨http://www.amy.hiho.ne.jp/okuhara/howtoj.htm⟩
[6] 橋本剛，前原彰太，川島哲也，小林康幸: 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，情報処理学会論文誌，Vol.54，pp.1930–1936，2013,
[7] R. Coulom, “Efficient selectivity and backup operators in Monte-Carlo tree search,” in *Proceedings of 5th International Conference on Computers and Games (CG 2006)*, 2006, pp. 72–83.
[8] L. Kocsis and C. Szepesvári, “Bandit based Monte-Carlo planning,” in *Proceedings of 17th European Conference on Machine Learning (ECML 2006)*, 2006, pp. 282–293.
[9] C. B. Browne, D. Whitehouse, P. I. Cowling, and S. Samothrakis, “A survey of Monte Carlo tree search methods,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 4, no. 1, pp. 1–43, 2012.
[10] R. Lorentz, “Using evaluation functions in Monte-Carlo tree search,” *Theoretical Computer Science*, vol. 644, pp. 106–113, 2016.
[11] M. H. M. Winands and Y. Björnsson, “Evaluation function based Monte-Carlo LOA,” in *Proceedings of 12th International Conference on Advances in Computer Games (ACG'09)*, 2010, pp. 33–44.
[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[13] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, “Mastering the game of Go without human knowledge,” *Nature*, vol. 550, pp. 354–359, 2017.
[14] K. Matsuzaki and N. Kitamura, “Do evaluation functions really improve Monte-Carlo tree search? — empirical analysis using Othello,” in *Proceedings of the 10th International Conference on Computers and Games (CG2018)*, 2018.