将棋における,評価値に基づくモンテカルロ木探索への Dynamic Komi の応用

竹内 聖悟[1,a)]

概要：モンテカルロ木探索は囲碁をはじめとするゲームにおいて成功を収め、組合せ最適化など幅広い応用が行われている。将棋ではモンテカルロ木探索を用いるプログラムの強さは評価関数とアルファベータ探索の組合せを用いるプログラムに及ばないが、将棋を対象としたモンテカルロ木探索の研究は近年増えている。
本稿では、プレイアウト中に評価関数及び静止探索を用いてプレイアウトの打切りを行う手法について、Dynamic Komi を利用し、打切りの基準となる評価値がその局面での真の minimax 値から外れている場合に起こる問題を解決する。その後、Dynamic Komi が持つ問題について逐次確率比検定や Accelerated UCT による改善手法を提案した。
モンテカルロ木探索将棋プログラムへ実装し、アルファベータ探索の結果を真の minimax 値とした収束についての実験や対戦実験を行った。実験結果から Dynamic Komi の有効性を示すことが出来たが、他の手法の有効性は確認できなかった。

Application of Dynamic Komi to Evaluation Value Based Monte-Carlo Tree Search in Shogi

SHOGO TAKEUCHI[1,a)]

Abstract: Monte-Carlo Tree Search (MCTS) becomes a popular search method in games, especially in the game of Go, and is also applicable in wide areas such as combinatorial optimization. In Shogi, the strength of MCTS programs is inferior to state-of-the-art programs which use a combination with an alpha-beta search and an evaluation function. However, the number of research for MCTS Shogi is increasing recently.
In our previous work, we proposed evaluation-value based MCTS methods and applied it to Shogi program. In this paper, we solve the problem with the method, suppose that a true minimax value is markedly different from the cutoff threshold, subsequently the most playouts result win and MCTS becomes unable to select the best move as a result. To solve this problem, we apply Dynamic Komi, which tunes the winning threshold according to the result of playouts. Additionally, we employ Sequential Probability Ratio Test and Accelerated UCT for improving Dynamic Komi.
We implement those methods for Monte-Carlo Tree Search Shogi program and measure Rooted Mean Squared Error between the minimax value and the the value by MCTS, and perform self-play tournaments. From the experimental results, we show the effectiveness of Dynamic Komi.

1. はじめに

コンピュータ囲碁の分野においてモンテカルロ木探索が多くのプログラムに用いられ、大きな成果を挙げている [9]。モンテカルロ木探索は General Game Playing[3] などゲームにおいても幅広く応用されている他、ゲームに限らず組合せ最適化など幅広い応用が試されている [2]。将棋では評価関数とアルファベータ探索の組合せが主流であり、モンテカルロ木探索はあまり成功していないが、将棋を対象とした研究も増えている [6][11][10][12]。

将棋へとモンテカルロ木探索を応用する利点としては、並列化が比較的容易と言われることや将棋では評価関数

[1] JST ERATO 湊離散構造処理系プロジェクト
JST ERATO Minato Discrete Structure Manipulation System Project
[a)] takeuchi@erato.ist.hokudai.ac.jp

の学習が成功しており、精度の高い評価関数をヒューリスティックとして利用可能であることや局面によっては、アルファベータ探索よりも正確な評価を行うことが可能であることなどが挙げられる。

モンテカルロ木探索における問題として、複数の手が勝ちである場合や全ての手が負けとなる場合に、最適な手を選ぶことができないという問題がある。複数の手が勝ちの場合、どの手を選んでも勝ちとなって差がつかず、最適な手を区別できない。また、全ての手が負けの場合もやはりどれを選んでも負けであるので、その中で最適な手を区別できない。この問題を解決するため、シミューレション（以下プレイアウトと呼ぶ）中に評価関数を使い、指手の中で評価値の高くなる手を選ぶようにする手法を提案した [11]。しかし、基準となる値がその局面の真の値から極端に離れている場合では、元のモンテカルロ木探索同様の問題が起こり、最適な手を選ぶことが出来ないという問題点がある。

本稿ではこの問題の解決方法として、Dynamic Komi[1] による解決を提案する。Dynamic Komi は囲碁のハンディキャップ戦での改善を動機として提案された、プレイアウト結果に応じて動的に勝利条件の閾値を変更する手法である。さらに、Dynamic Komi におけるコミの更新について逐次確率比検定 [7][13], コミの異なるプレイアウト結果が混同することについて Accelerated UCT[4] の利用による改善を提案する。最後に、各手法をモンテカルロ木探索将棋プログラムへと応用し、基準となる値と真の値の差の測定、対戦実験を行い、提案手法の有効性を示す。

2. 関連研究

本研究に関連のある、評価値を利用したプレイアウトと Dynamic Komi, 逐次確率比検定 について説明を行う。

2.1 評価値を利用したプレイアウト

プレイアウト中に評価関数を使う手法は Lines of Action や [8], Amazons[5] で提案され、成功している。前者は閾値との大小比較から勝敗を決定するために、後者は葉局面での勝敗の値として、それぞれ評価値を利用する。将棋においては、両方の手法に静止探索を組合せ、更にモンテカルロ木探索のルート局面での静止探索の値からの大小で勝敗を決定する手法が竹内らによって提案されている [11]。本稿では、評価値に応じてプレイアウトを打ち切る前者の方法を用いる。

竹内らの手法では、モンテカルロ木探索のルート局面における静止探索の値 $V_{qs,root}$ に対して一定のマージン m を取り、プレイアウト中の各局面で $V_{qs,root} \pm m$ との大小から勝敗を決定する。本稿では、基準となる値をルート局面の静止探索の値 $V_{qs,root}$ から拡張し、評価値 V_b とした手法を用いる。この手法により得られる勝敗は、局面 P に対して、次のような関数 $MC(P)$ として書ける。

$$MC(P) = \begin{cases} +1 & (V(P) > V_b + m) \\ -1 & (V(P) < V_b - m) \\ MC(Next(P)) & otherwise \end{cases}$$

なお、$V(P)$ は局面 P における評価値で、実際には null-window search によって大小の比較だけが行われる。$Next(P)$ はプレイアウトの次の局面を返す関数である。

通常のモンテカルロ木探索においては、複数の手が勝ちであったり、全てが負けの手となる場合に、正しく勝率の高い手を選べなくなる問題があった。この評価値を利用した手法では、評価値がルート局面よりも高くなるような手を選ぶため、勝ちや負けの局面でも比較的良い手を選ぶと考えられ、実際に対戦実験では、この手法を採用したプログラムが有意に勝ち越した [11]。

しかし、この手法においても、V_b と真の minimax 値 V^* が大きく離れている時に勝率の高い手を正しく選べなく成る問題がある。例えば、ある局面の V^* が $V_b + m$ よりも十分大きく、複数の手の勝率が 1 に近くなる場合で、この時はそれらの手で最適なものを区別できず、最適な手を選択できない。また、V^* が $V_b - m$ を大きく下回る場合も同様で、全ての手の勝率が 0 に近くなり、最適な手を選べない。

2.2 Dynamic Komi

囲碁において、勝ちや負けに近い局面で指手を正しく選択できなくなる問題、特にハンディキャップ戦において起こる問題を解決するために、Dynamic Komi が提案された [1]。囲碁では、先手の目数と後手の目数との差がコミより大きければ先手の勝ちとなる。先手が勝っている局面を例とすると、コミを大きくすることで幾つかの手が先手勝ちでなくなる。それらの手よりも、残った手の中にある先手勝ちの手は良い手であり、正しい手を選択しやすくなる。

論文中では LinearHandicap, ScoreSituaional, ValueSituaional の 3 つの手法が提案されている。LinearHandicap はハンディキャップを初期局面から手数に応じて線形に減少させる手法で、ScoreSituational は一定量のプレイアウトにおける平均目数に応じてコミを増加させる手法であり、ValueSituational は一定量のプレイアウトにおける勝率を見て、コミを増減させる手法である。ValueSituational では一度でもコミを下げたとすると下がったコミを Ratchet (歯止め) として、それ以降の更新では、Ratchet より大きな値になることはない。ただし、コミが負の値の時はこの Ratchet を設定しない。コミの更新のタイミングについては、探索中に更新する online komi と探索開始時に前回の結果を元に更新する offline komi の 2 種類があるが online が良いという結果がある。

Baudiš は考えられる問題として、コミの異なるプレイア

ウト結果が混在することを挙げているが、実験的には問題がないと述べている。

LinearHandicap はハンディキャップが事前に分かっている必要がある上、手数に応じて減少していくので、本稿の手法に適さない。ScoreSituational はプレイアウト末端において正確な評価値が必要となるが、今回の手法は末端において正確な評価値を求めないため、これも適さない。

以上の理由から ValueSituaional を採用する。また、更新には online komi を利用する。なお、囲碁においてコミは 1 目単位であるが、本稿の手法においては評価値を対象とするため、適当な値を探す必要がある。本稿では、初期値として歩 1 枚の値を用い、増減の向きが逆転する度に半減させるというヒューリスティックを用いる。Ratchet については、上記のヒューリスティックの利用により同様の効果が得られることを期待し、今回は採用しない。

2.3 逐次確率比検定

逐次確率比検定 (Sequential Probability Ratio Test)[7][13] とは、事前にデータ数を定めずに逐次的に検定を行う手法である。同じデータに対して検定を繰り返し行うことは、第一種の過誤の確率を増加させるため、単純な検定の繰り返しは避ける必要があるが、逐次確率比検定では事前に定めた第一種、第二種の過誤となる確率を満たしながら、逐次に検定を行うことが出来る。また、他の利点として、検定に要するサンプルサイズを平均的に少なくなることがある。逐次確率比検定は、品質管理工学や医学診断などの分野で用いられる。

手法としては、最初に 2 つの帰無仮説を H_0, H_1 を立てる。例えば、H_0 はプレイヤ A とプレイヤ B とのレーティングの差が 0.0, H_1 はプレイヤ A とプレイヤ B とのレーティングの差が 5.0 という仮説が考えられる。θ を仮説のもとでの確率分布パラメータとして、k 回目の試行から得たサンプルを x で表記する。$p_\theta(x)$ は θ の元で x が起こる確率を表す。すると、もっともらしさを表す尤度 (Likelihood) と、各 $\theta_{\{0,1\}}$ での尤度比 (Likelihood Ratio) は次式のようになる。

$$L_k(x,\theta) = \prod_{i=1}^{k} p_\theta(x_i)$$

$$LR_k(x,\theta_1,\theta_0) = \frac{L_k(x,\theta_1)}{L_k(x,\theta_0)}$$

逐次確率比検定では、パラメータ α,β の元で、

$$LR_k(x,\theta_1,\theta_0) \leq \frac{\beta}{1-\alpha}$$

$$LR_k(x,\theta_1,\theta_0) \geq \frac{1-\beta}{\alpha}$$

上記の条件を満たす時に、それぞれ H_0, H_1 を採択し、それ以外の条件

$$\frac{\beta}{1-\alpha} < LR_k(x,\theta_1,\theta_0) < \frac{1-\beta}{\alpha}$$

を満たす時は次の標本を得るとする。すると、第一種、第二種の過誤となる確率を α,β で抑えることができる [7]。なお、実際に計算する際には、乗算を避けるために対数を取った値で計算する。

オープンソースのチェスソフト Stockfish[*1] では、開発テストのフレームワーク[*2] の中で、変更によるレーティング向上の有意性を検定するために、逐次確率比検定を利用している。勝敗の分布については、レーティングに基づいて計算する。なお、実際に Elo レーティングで何点向上したかを確認する際には、固定回数の対戦から計測している。

以下では、Stockfish の開発フレームワークにおける逐次確率比検定の尤度比の計算を記す。勝ち, 負け, 引分をそれぞれ W, L, D として表し、全体に占める割合は $R(X)$ で表す (X は W, L, D のいずれか)。以下では、いずれも 0 でないとして計算する。まず、引分の影響を計算する。

$$\mathit{drawelo} = 200.0 \times \log_{10}\Big(\frac{(1-R(W))\times(1-R(L))}{R(W)\times R(L)}\Big)$$

引分の影響を加味し、レーティング elo から勝ち負け引分の分布を次のように計算する。

$$p(W) = \frac{1}{1+10^{(-elo+drawelo)/400.0}}$$

$$p(L) = \frac{1}{1+10^{(elo+drawelo)/400.0}}$$

$$p(D) = 1 - p(W) - p(L)$$

この結果から尤度は次のように計算され、

$$LR_k(x,\theta_1,\theta_0) = \prod_{i=1}^{k} \frac{p_1(x_i)}{p_0(x_i)}$$

対数尤度比 (Log-Likelihood Ratio) が次のように求まる。

$$\begin{aligned} LLR_k(x,\theta_1,\theta_0) &= \sum_{i=1}^{k} \ln\Big(\frac{p_1(x_i)}{p_0(x_i)}\Big) \\ &= \sum_{X=\{W,L,D\}} \#X \times \ln\Big(\frac{p_1(X)}{p_0(X)}\Big) \end{aligned}$$

2.4 Accelerated UCT, Discounted UCB

Discounted-UCB[*3], Accelerated UCT[4] について説明する。

前者は過去のプレイアウトの寄与を小さくする手法で、後者はモンテカルロ木探索のゲーム木探索部分で、探索時に選択された手以外の寄与を小さくする手法である。前者は、単純に古いプレイアウトに対して、あらかじめ定めた定数をかけて寄与を小さくしていく手法である。後者は、UCT の木を辿り、プレイアウトの結果をルートに戻してい

[*1] stockfishchess.org
[*2] http://tests.stockfishchess.org/tests
https://github.com/glinscott/fishtest
[*3] https://www.lri.fr/~sebag/Slides/Venice/Kocsis.pdf

く際に、UCB 値が最大であった、選ばれた手については寄与はそのままとして、それ以外の手の寄与を小さくする手法である。

Hashimoto らは囲碁プログラム Fuego へと両方の手法を用いて対戦実験を行った。Discounted-UCB は元の fuego に対して勝ち越さず、Accelerated UCT は有意に勝ち越す結果となり、Accelerated UCT の有効性を示した [4]。

3. 提案手法

評価値に応じてプレイアウトを打ち切る手法の問題を解決するために Dynamic Komi を用いた手法の提案を行い、さらなる改善のために逐次確率比検定や Accelerated UCT を用いる手法について提案する。

評価値に応じてプレイアウトを打ち切る手法における問題とは、打切りの基準となる値 V_b よりも十分に良い手が複数あるか、全ての手が V_b よりも十分に悪い時には、それらが全て最適な手の候補となって最適な手を選べなくなるという問題である。

3.1 Dynamic Komi の適用

評価値に応じてプレイアウトを打ち切る手法に対し、Dynamic Komi を応用し、基本となる評価値 V_b を動的に調整し、問題を解決する手法を提案する。最適な手を選べない問題において Dynamic Komi を考えると、V_b よりも良い手が複数ある場合は勝率が高くなり V_b が大きく、全ての手が V_b よりも悪い場合には勝率はほぼ 0 に近くなるため V_b が小さくなり、Dynamic Komi によって V_b が適切な値へと近づき、最適な手を選べるようになると期待される。

従来手法では、V_b は局面毎に静止探索を行った値を使っていたが、Dynamic Komi では調整された V_b を次の手番でも引き継いで使う。また、2.2 節で述べた理由から、一定回数以上のプレイアウトにおける勝率からコミの増減を決める ValueSituational 及び、探索中に更新する online komi を採用するが、Ratchet は採用しない。

一方、Dynamic Komi における問題として、基本となる評価値 V_b の調整前後のプレイアウト結果が同等に扱われる問題点があり、Baudiš は問題ないと述べている [1] が、Discounted-UCB や Accelerated UCT[4] の利用により、評価値 V_b を調整する前の結果の寄与を小さくできるため、改善が期待できる。

コミの更新のタイミングや増減値については、試行錯誤で決められている。今回、増減値は歩 1 枚を初期値として、増減が逆向きとなる度に半減させるヒューリスティックを採用しているが、プレイアウトの勝率やレーティングに応じて値を変えるなどの案があるが、今回は扱わない。更新のタイミングについての工夫を、次節で述べる。

3.2 逐次確率比検定の利用

Dynamic Komi におけるコミの更新について考えると、勝率が閾値以上かどうかだけで判断しているが、これを検定として考えると逐次検定だとみることができる。よって、そこに逐次確率比検定を利用することで、第一種、第二種の過誤となる確率を定めることができ、平均的なプレイアウト回数を減少させることも期待できる。

本稿では、2.3 節で述べた Stockfish の開発フレームワークにおけるレーティング計算を利用する。レーティングを採用する理由は、勝率 $0.55, 0.95$ において勝率の差 0.01 の効用は異なると考えられるが、レーティングであれば、それらを加味した計算になっていると考えられるためである。プレイアウトはランダムなゲームであるので、その勝敗からレーティングを計算することができ、適用も容易である。勝ち（負け）の時に、レーティングが正（負）の閾値より小さい（大きい）ことが帰無仮説となり、帰無仮説が棄却される時にプレイアウト打切りの閾値 V_b を変動させる。

逐次確率比検定の利用による利点として、過誤となる確率の制御と更新までのプレイアウト数が少なくなることがあるが、従来の Dynamic Komi は一般的な検定を行うわけではないので単純な比較はできない。

4. 実験

提案手法の有効性を示すため、Dynamic Komi (以下 dynk), 逐次確率比検定 (以下 sprt), Accelerated UCT (以下 acc) の実装と実験を行った。なお、Discounted-UCB については、元のプログラムに実装したが、Discounted-UCB を利用するプログラムが有意に負け越したため、以下の実験では利用していない。

実験用の将棋プログラムには、GPS 将棋[*4](rev.2781, Open Shogi Library[*5] rev.4504) を元に作成したモンテカルロ木探索プレイヤを利用した。竹内らの手法 [11] をベースとし、sato ら [6] の工夫を取り入れている。

4.1 真の minimax 値への収束

提案手法により V_b が真の minimax 値へと近づいているかを確認するための実験を行った。真の minimax 値は不明であるので、GPS 将棋のアルファベータ探索の結果を近似値 V^* として利用する。V_b の値には静止探索の値を利用する。これは Dynamic Komi や逐次確率比検定 での V_b の初期値となる。GPS 将棋のアルファベータ探索では、およそ深さ 12 に相当する limit 1,200 で探索を行い、モンテカルロ木探索プログラムでは、プレイアウト数を 30,000 とし、3,000, 10,000, 30,000 毎にデータを取った。局面は、棋譜の

[*4] http://gps.tanaka.ecc.u-tokyo.ac.jp/gpsshogi/
[*5] http://gps.tanaka.ecc.u-tokyo.ac.jp/gpsshogi/pukiwiki.php?OpenShogiLib

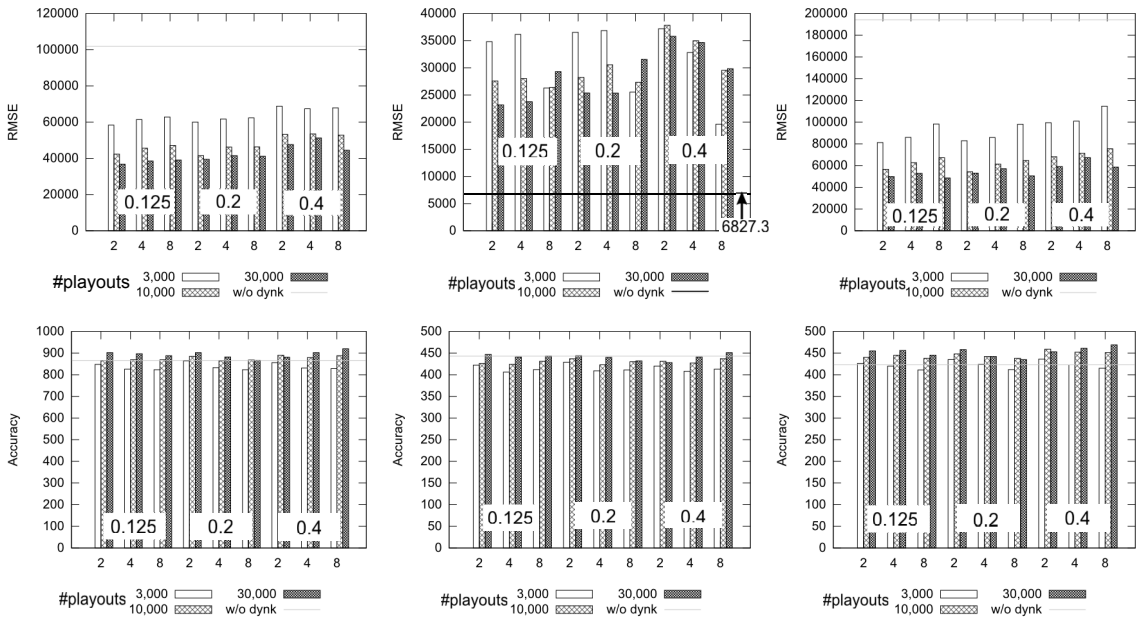

図 1　Dynamic Komi の RMSE と一致数 (上: RMSE, 下: 一致数, 左: 全データ, 中央: 差が歩 1.5 枚未満, 右: 差が歩 1.5 枚以上)

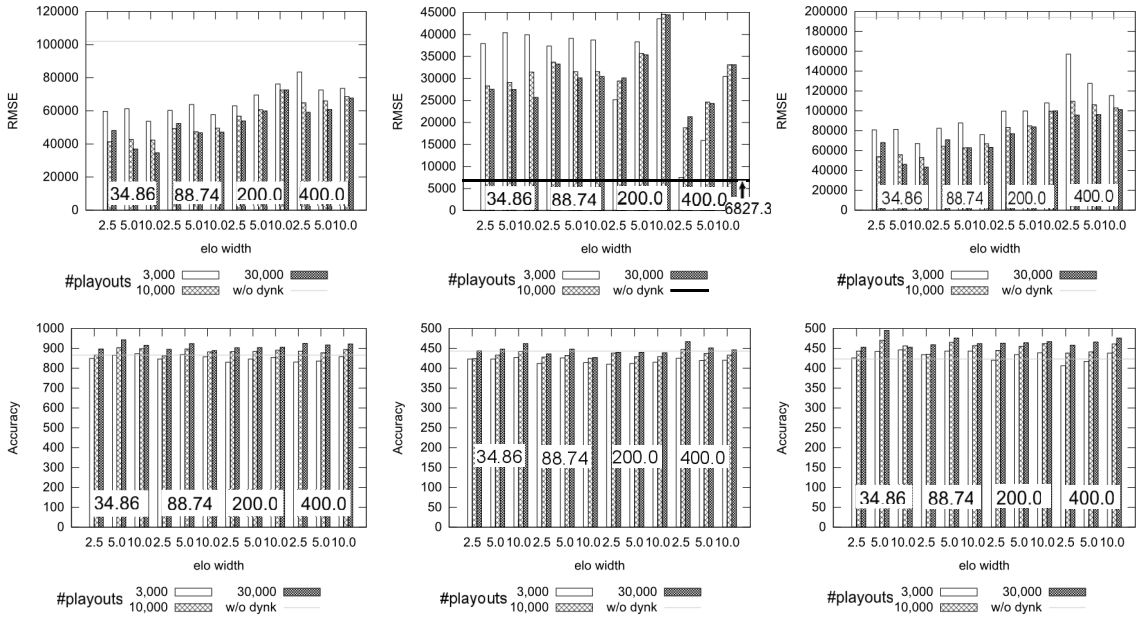

図 2　逐次確率比検定による Dynamic Komi の RMSE と一致数 (上: RMSE, 下: 一致数, 左: 全データ, 中央: 差が歩 1.5 枚未満, 右: 差が歩 1.5 枚以上)

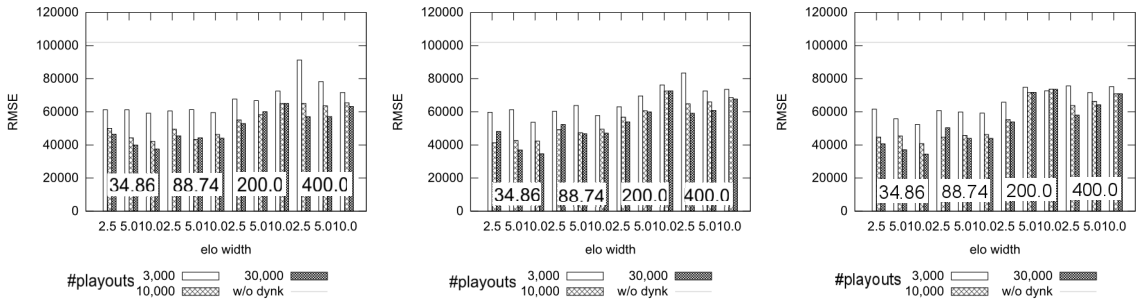

図 3　逐次確率比検定による Dynamic Komi のエラー率毎の RMSE ($\alpha = \beta =$ 左: 0.001, 中央: 0.01, 右: 0.05)

30 手目から終了の 15 手前までで詰や必死、王手のかかっていない局面を対象として、ランダムに 1,500 局面を選択したが、3 局面はモンテカルロ木探索 によって詰や必死が発見されたため、1,497 局面が対象となった。

$|V_b - V^*|$ の二乗平均平方根 (Rooted Mean Squared Error, 以下 RMSE) 及びアルファベータ探索との指手の一致数を得た。また、V_b の初期値が V^* から離れているケースとそうでないケースを見るため、差が歩 1.5 枚より大きいかそれ以下であるかでデータを分けた。結果を図 1, 2 にまとめた。なお、逐次確率比検定 では、過誤となる確率が 0.01 のものを掲載しているが、確率による大きな影響はなかったため、確率を変えた全体での RMSE の図 3 で傾向を述べることとして、他の図では 0.01 の結果を利用した。

図 1 の X 軸は、プレイアウト回数の閾値を表し、数値の 100 倍以上のプレイアウト後に勝率の閾値を超えるかを調べ、その後 100 プレイアウト毎にチェックを行う。グラフを横断する数値 $\{0.125, 0.2, 0.4\}$ はプレイアウトの勝率の閾値を表し、$0.5 \pm \{0.125, 0.2, 0.4\}$ 以上/以下の時にコミの更新を行う。

図 2 のグラフを横断する数値 $\{34.86, 88.74, 200.0, 400.0\}$ はプレイアウトのレーティングの閾値を表し、X 軸の $\{2.5, 5.0, 10.0\}$ は検定に用いるレーティング幅を表し、$\{34.86, 88.74, 200.0, 400.0\} \pm \{2.5, 5.0, 10.0\}$ 以上/以下の時にコミの更新を行う。前述の通り、過誤となる確率 $\alpha = \beta = \{0.05, 0.01.0.001\}$ を試しているが、この図では 0.01 のみを示している。図 3 では RMSE について結果を示しているが、全体的には過誤となる確率が大きい方が RMSE は小さくなっている。一方で指手の一致数では大差がないため、0.01 の値のみを以下では利用する。

全局面を使った結果（図 1, 2 左）を見ると、RMSE が小さくなっており、Dynamic Komi によって真の minimax 値へと近づくことが示された。また、指手の一致数も増加しており、提案手法の有効性を確認できる。また、逐次確率比検定 による Dynamic Komi は RMSE では Dynamic Komi と同等だが、指手の一致数では上回るなど、有効性が期待できる。

一方、歩 1.5 枚未満の局面では、Dynamic Komi を利用しない手法よりも悪い結果となった。一方で、指手の一致度について見ると、図 1 中央下や図 2 中央下のように、Dynamic Komi なしと同等かそれ以上の結果を得ているものもあり、指し手として悪い手を選択するようになったわけではないことがわかる。

4.2 対戦実験

各手法におけるパラメータの効果や、提案手法の有効性を示すために、対戦実験を行った。

評価値に基づくプレイアウト打切りとモンテカルロ木探索を実装した将棋プログラム (以下 orig) の他、Dynamic Komi を適用したプログラム (dynk), Dynamic Komi に逐次確率比検定を適用したプログラム (sprt), Accelerated UCT (acc) を実装したプログラムを用意した。

表 1　対戦実験 (勝-分-負) dynk 対 orig : 3,000 プレイアウト, 1,000 局

閾値＼回数	2	8
0.125	651-0-349	608-0-392
0.400	620-1-379	615-1-384

表 2　対戦実験 (勝-分-負) sprt 対 orig, エラー率 0.01: 3,000 プレイアウト, 1,000 局

elo ＼幅	2.5	10.0
34.86	650-2-348	618-1-381
88.74	648-1-351	620-0-380
400.0	466-1-533	529-1-470

以下の対戦では、1 手あたりのプレイアウト数を 3,000 として、初期局面には実際の棋譜から 30 手進めた局面を 500 局用意し、先後を入れ替えて 1,000 局行った。プレイアウト数を固定した対戦であるので、性能の異なるマシンを利用して対戦を行っている。持ち時間のある対戦を行う際にはマシンを統一する。

4.2.1 Dynamic Komi

Dynamic Komi の有効性及び、パラメータの効果を調べるため、パラメータを変えて Dynamic Komi なしのプログラムとの対戦を行った。

結果を表 1 にまとめた。全て有意水準 5%で有意に勝ち越し、Dynamic Komi の有効性を確認できた。有意な差ではないが、更新の起こりやすい方が勝率が高い傾向があるように考えられる。

4.2.2 逐次確率比検定を用いた Dynamic Komi

逐次確率比検定 を用いた Dynamic Komi について有効性を確認するため対戦を行った。過誤となる確率を 0.01 に固定し、プレイアウトの勝率に関する閾値, 閾値を超えた割合などを変動させて対戦を行った結果を表 2 にまとめた。

エラー率や検定するレーティングは大きいほど、検定の際のレーティングの幅は小さいほど、更新は起きやすくなり、minimax 値への収束の実験では更新頻度が高いものほど、RMSE は小さくなる傾向があった。レーティング 400.0, 幅 2.5 の最も更新が起こりにくいプログラムは有意水準 95%で有意に負け越し、レーティング 400.0, 幅 10.0 では有意な差が得られなかったが、他のパラメータセットは有意水準 95%で有意に勝ち越した。Dynamic Komi の時と同程度であり、逐次確率比検定 による優位性は見られなかった。また、Dynamic Komi の時と同様にコミの更新が多く起こるようなパラメータセットが有効な傾向が見られる。

レーティング 400.0, 幅 2.5 のプログラムが負け越した原因としては、更新が起こりにくいため、V^* が急増や急減した場合に V_b の更新が追いつかずに、静止探索を行う orig

表 3　対戦実験 (勝-分-負) sprt 対 dynk: 1,000 局

	3,000 プレイアウト	10 分切れ負け
sprt 対 dynk	504-3-493	496-2-502

表 4　対戦実験 (勝-分-負) sprt+acc 対 sprt : 3,000 プレイアウト, 1,000 局

k	4	5	6	7
sprt+acc	513-1-486	497-2-501	485-1-514	514-485

の方が正確な V_b を利用して探索できることなどが考えられる。

Dynamic Komi からの改善を確認するために、dynk と sprt の対戦を行った。dynk は勝率の高かった勝率の閾値 0.25, 回数の閾値 2 、sprt はレーティング 88.74, 幅 2.5, 過誤となる確率 0.01 を用いた。これまで同様の 3,000 プレイアウトの対戦に加え、10 分切れ負けでの対戦を行った。結果を表 3 にまとめたが、有意な差はなかった。

4.2.3 Accelerated UCT

Accelerated UCT による改善を確認するため、を sprt に加えたものと sprt との対戦を行った。結果は表 4 のようになり、勝ち越すものもあったが有意な結果は得られなかった。なお、sprt のパラメータはレーティング 88.64, 幅 2.5, 過誤となる確率 0.01 を利用した。表中の k は Accelerated UCT 中の寄与の逓減に関するパラメータで、探索中に選ばれなかった手の寄与は $1.0 - 10^{-k}$ 倍となる。

5. 議論

4.1 節で得られたように、真の minimax 値と打切り基準値が近い（実験では歩 1.5 枚未満）場合、Dynamic Komi によって調整された V_b との RMSE は静止探索による初期値よりも悪くなる傾向がある。一方で指手の一致度は大きく下がることはなく、むしろ向上しているものもあり、RMSE が大きいからと言って一概に悪いと言えないことがわかる。原因としては、V_b は打切りの閾値に使われる値であるため、Dynamic Komi によって調整された値はプレイアウト打切りに適した値ではあるが、評価値としては正確ではない、ということが考えられる。Dynamic Komi の研究中ではコミの更新を行わない勝率の範囲を $(0.45 : 0.5)$ としており、少し自分が不利な局面にロックするパラメータが良いとしており [1] 、モンテカルロ木探索に適した V_b と正しい評価値とが異なることが示唆されている。

今回解決しようとした問題は、複数の指手が V_b よりも十分に大きい場合と全ての指手が V_b よりも十分に小さい場合の二通りで、複数と全ての手であるため非対称であった。今回の勝率やレーティングの閾値では対象な値を利用したが、非対称な値を用いることによる改善が考えられる。実際、前段落にあるように Baudiš らは非対称な閾値を利用している。

Dynamic Komi によって調整された V_b が評価値としての正確さを求めたものでないように、探索の目的自体、正しい minimax 値を得ることではなく、最善手とそれ以外の手を区別することである。UCT では最善の手に多くの探索資源を割り当てる傾向があり、モンテカルロ木探索が進むに連れて、有利な局面であれば勝率が高くなるのは自然な結果である。単に勝率が高いからとコミを上げるのではなく、最善手とそれ以外の手を区別して更新する必要があり、単純には全てのプレイアウトが最善手の後で行われた場合には Dynamic Komi を行わない、最善の手が負けた割合と次善手または最善以外の手で勝った割合について比較を行うなどの対処が考えられる。

今回はコミの増減量については、方向が逆転する毎に半減させるヒューリスティックを用いた。逆方向の結果が得られた後に、続けて同じ方向へ更新することで元の値を超える/下回ることは起こり得るので、徹底するのであれば Ratchet を導入することが考えられる。また、逐次確率比検定 では検定を行っているので、一度ある値以上または以下であると検定されたのであれば、それ以降はその逆の結果が得られては検定の意味がなく、検定結果に応じて Ratchet を利用することで、改善の可能性が考えられる。

今回逐次確率比検定による改善は、対戦では有意な結果を得られなかったが、指手の一致数では上回っており、多少の優位性を示すことが出来た。一致数は 30,000 プレイアウトでの結果を比較しており、対戦では 3,000 プレイアウトと少なかったことも原因として考えられる。また、逐次確率比検定 を利用するアイデアは、幅広く応用が可能であると考えている。例えば、反復深化は複数回の検定を行っていると考えることができ、逐次確率比検定を適用することで改善できる部分があると期待している。

6. 終わりに

本稿では、プレイアウト中に評価関数及び静止探索を用いてプレイアウトの打切りを行う手法 [11] について、打切りの基準となる評価値が真の値から外れている場合に起こる問題について述べた。その後、Dynamic Komi[1] の利用による解決方法を提案し、さらに、Dynamic Komi におけるコミの更新に逐次確率比検定を適用する手法や Accelerated UCT の利用による改善を試みた。

真の minimax 値への収束について実験を行い、Dynamic Komi により、minimax 値との誤差が小さくなることを確認し、提案手法により問題点が解決できることを示した。また、逐次確率比検定による手法は指手の一致数の点で Dynamic Komi を上回っており、有効性が期待された。

対戦実験から Dynamic Komi によって有意に勝率が向上することを確認し、提案手法の有効性を示した。一方で、逐次確率比検定や Accelerated UCT による有意な改善は確認できなかった。

今後は、5 章で述べたような、Dynamic Komi を改善する

アイデアの実装及び実験、議論した内容を確認するための実験などを行っていく予定である。mた、逐次確率比検定による Dynamic Komi の改善手法は将棋に特化した手法ではないため、Dynamic Komi が提案された囲碁においても有効であると考えられ、囲碁プログラムで実装、実験することも考えている。

参考文献

[1] Petr Baudiš. Balancing mcts by dynamically adjusting the komi value. *ICGA Journal*, Vol. 34, No. 3, 2011.

[2] C.B. Browne, E. Powley, D. Whitehouse, S.M. Lucas, P.I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 1, pp. 1–43, march 2012.

[3] Hilmar Finnsson and Yngvi Björnsson. Simulation-based approach to general game playing. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence (AAAI-2008)*, pp. 259–264, 2008.

[4] Junichi Hashimoto, Akihiro Kishimoto, Kazuki Yoshizoe, and Kokolo Ikeda. Accelerated uct and its application to two-player games. In H.Jaap Herik and Aske Plaat, editors, *Advances in Computer Games*, Vol. 7168 of *Lecture Notes in Computer Science*, pp. 1–12. Springer Berlin Heidelberg, 2012.

[5] Richard J. Lorentz. Amazons discover monte-carlo. In *CG '08: Proceedings of the 6th international conference on Computers and Games*, pp. 13–24, Berlin, Heidelberg, 2008. Springer-Verlag.

[6] Yoshikuni Sato and Reijer Grimbergen. A shogi program based on monte-carlo tree search. *ICGA Journal*, Vol. 33, No. 2, pp. 80–92, 2010.

[7] A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, Vol. 16, No. 2, pp. pp. 117–186, 1945.

[8] Mark H. M. Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. Monte carlo tree search in lines of action. *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 2, No. 4, pp. 239–250, 2010.

[9] 美添一樹. モンテカルロ木探索 – コンピュータ囲碁に革命を起こした新手法. 情報処理, Vol. 49, No. 6, pp. 686–693, 2008.

[10] 関栄二, 三輪誠, 鶴岡慶雅, 近山隆. 将棋におけるモンテカルロ木探索の特性の解明. ゲームプログラミングワークショップ 2012 論文集, 第 2012 巻, pp. 68–75, nov 2012.

[11] 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. ゲームプログラミングワークショップ 2010 論文集, 第 2010 巻, pp. 86–89, nov 2010.

[12] 横山大作. モンテカルロ木探索アルゴリズムの将棋への適用. ゲームプログラミングワークショップ 2012 論文集, 第 2012 巻, pp. 76–83, nov 2012.

[13] 松原望. ベイズ統計学概説：フィッシャーからベイズへ. 培風館, 2010.