モンテカルロ木探索アルゴリズムの将棋への適用

横　山　大　作[†1]

モンテカルロ木探索は囲碁などで有効な探索とされているが、正確な先読みを必要とする性質を持つ将棋においては適用が難しい。我々は、(1) 乱数を付加した評価値を用いた木探索をプレイアウトとする (2)bayesian approach による評価値伝搬を利用する、という 2 つの手法を用いたモンテカルロ木探索を提案し、将棋への適用を試みた。対戦による評価の結果、利用リソース量の増加に伴い勝率向上が得られることが確認された。

An Application of Monte-Carlo Tree Search Algorithm for Shogi

DAISAKU YOKOYAMA[†1]

Monte-Carlo Tree Search (MCTS) algorithm is quite effective for playing Go, however it has some weakness for playing tactical games, like Shogi. We propose a new MCTS method that consists of (1) adding random value for evaluation of leafs of a game tree, and treating the tree as a playout; (2) applying bayesian approach to propagate distributions of leaf values. We apply it for Shogi player. Evaluation by self-fight shows that the proposed method obtains better performance when it gets more resources.

1. はじめに

1.1 背　　景

将棋、囲碁などに代表される 2 人ゲームは、相異なる 2 つの目的関数を持つプレイヤが互いの利益を最大化しようとする複雑な構造を持つ探索問題である。人間が楽しむエンタテイメントの目的のみならず、人工知能分野に応用される最適化手法や、極めて大規模な解空間を効率よく扱う分散探索計算技術の応用分野として、広く研究されてきた。オセロ、チェス、将棋などは、局面の優位度を数値化する評価関数を用いたミニマックス木探索により次の指し手を求めることが通例であり、現時点ではこの手法が最良 (最強) の結果を出している。探索空間の広さなどの要素により、オセロとチェスではコンピュータが人間を越えた強さを持つが、将棋に関しては現在人間のトッププレイヤの強さには届いておらず、今後数年から 10 年程度の間に人間を追い越すことを目標とした研究が行われている。

一方、囲碁に関しては従来の木探索アルゴリズムでは人間よりはるかに弱いプレイヤしか構築できなかったが、乱数を用いたモンテカルロ木探索を利用したアルゴリズムが提案され[1]、急速に強さが向上している。これは、モンテカルロ木探索が比較的容易に分散計算化可能なため、コンピュータリソースを大量に利用して高速化できることも大きく影響している。この新たなアルゴリズムの成功を受けて、将棋においてもモンテカルロ木探索の適用が試みられてきたが、従来の木探索手法より良好な性能は得られていない[2)3)]。モンテカルロ木探索では、乱数を用いた適当な手順により終局まで局面を進める「プレイアウト」を多数繰り返して評価を行うが、ゲームの性質上、囲碁と異なり将棋では正しいプレイアウトを生成することが困難であること、および多数のプレイアウトによる評価が通常の木探索の正確さに及ばないという事情に依ると考えられている。特に後者については、ある局面において正解といえる手順がごく少数に限られ、その正解手順を長期間たどり続けて初めて局面に優劣が付くようなゲームに関して、現在のモンテカルロ木探索手法では効率よく探索空間を絞ることができないという問題点が露呈しているといえる[4)]。

また近年、将棋において分散計算を適用するために「合議アルゴリズム」が提案された[5)]。これは局面評価関数に乱数を加えた複数のプレイヤが示す次の一手を単純に多数決するというだけのものであるが、いくつかの将棋プログラムで確かに強くなることが検証されている。しかし、逐次アルゴリズムと同一計算リソースを用いての比較は行われておらず、以前から研究さ

†1 東京大学生産技術研究所
Institute of Industrial Science, The University of Tokyo
yokoyama@tkl.iis.u-tokyo.ac.jp

れてきた分散探索アルゴリズムとの比較も含めて、どの程度の有効性を持つものであるかは検討の余地がある。また、適用されてきたのは 10 並列以下程度の場合のみであり、より大規模な分散計算環境では有効性が落ちることが予想されてもいる[⋆1]。また、モンテカルロ探索とのアルゴリズム的類似性もあると予想されるが、詳細な検討は存在していない。

1.2 本研究の目標

従来のプレイアウトを利用したモンテカルロ探索は、将棋では有効に機能していない。そこで、本論文では、将棋を対象としたゲームプレイヤにおいて従来用いられてこなかった、モンテカルロ探索を指向したゲーム木探索手法を提案し、その有効性を検証することを目指す。モンテカルロ探索を利用するために、本研究では、従来用いられなかった以下の 2 つの技術要素を組み合わせることを提案する。

- 評価関数に乱数を与えることでモンテカルロ探索のプレイアウトを構成する
- bayesian approach を用いることで評価値分布を反映したモンテカルロゲーム木を構築する

本論文では、この 2 つを用いたコンピュータプレイヤを実装し、アルゴリズムに類似性が高い従来技術である合議法と比較してその有効性を評価する。

2. 関 連 研 究

2.1 モンテカルロ木探索での評価関数の利用

モンテカルロ木探索は、単純に乱数による試行を繰り返すモンテカルロ法を、min-max 木の探索と組み合わせたものであり、選択的に成長させた min-max 木に従ってゲームの最善手を求める手法である[6]。モンテカルロ探索を行う範囲を、探索途中に得られている結果をもとに絞り込み、有望なゲーム空間に対してより詳細にランダムな探索を試みることで、高速に探索結果を収束させることを可能にしている。近年、UCB(Upper Confidence Bound 値) を利用して絞り込みを行う UCT 探索が提案され[7]、特に囲碁において良好な性能を示し、広く使われ始めている[1]。

モンテカルロ木探索はランダムな手順により勝ち負けが確定するまで局面を進める (プレイアウト) ことで評価を行うため、局面の有利さを数値化する評価関数を用いる必要がない。このため、囲碁のように精度の良い評価関数を作ることが難しい問題で特に有望な手段として適用が始まった。その後、Amazons[8] や Lines of Action(LOA)[4] など、ある程度信頼できる評価関数が得られているゲームにも適用が試みられている。

Winands ら[4] は、プレイアウト中に評価関数を用いて局面評価を行い、ある閾値以上の優劣がついた時点で勝敗が確定したと判断する手法、評価値に従って手の選択確率を変化させる手法、深さ 1 の探索を繰り返すグリーディな手法、を組み合わせて利用する手法を提案し、LOA に適用した結果、従来のゲーム木探索によるプレイヤとの対戦実験において 46%の勝率を得た。従来の評価関数を用いた探索で十分な強さを実現している LOA というゲームにおいて、同等の強さをモンテカルロ木探索で実現したことになる。

このように、良好な評価関数を持つ問題領域において、モンテカルロ木探索にその知識を利用しようとする研究は始まっている。しかし、従来型の木探索が有効な問題領域では、評価関数以外にも探索そのものに関する多数の技術が確立されている。例えば将棋においては、futility-pruning や実現確率などの枝刈り技法、詰み専用探索の利用など、様々な工夫を利用して高性能な木探索が実現されている。しかし、このような木探索の既存技術をモンテカルロ木探索において利用しようという試みはまだ行われていない。我々は、プレイアウトの代わりに従来の木探索をそのまま利用できるアルゴリズムを提案し、モンテカルロ木探索において従来技法の効果をより有効に活用することを目指している。

モンテカルロ木探索には、明確な勝ち負けが確定している局面 (いわゆる「詰み」の局面) でも、そのことがわからないという課題点がある。これは、ある局面の値が確定している場合でも、その「確定している」という事実が木探索の中で扱えていないことに起因する。これに対し、プレイアウトによる勝率の値とは別に、勝ち負けが確定した局面については特別な評価値を与え、min-max 木によって伝搬させることで詰み局面を扱う、Monte-carlo Tree Search Solver[9] という技法が提案されている。これにより問題点への対処は可能であるが、モンテカルロ木探索で利用するものとは別のアルゴリズムを併用する必要があるため、複雑さは増してしまう。我々は、プレイアウトに相当する探索によって得られる評価値の分布をそのまま扱える手法を提案する。我々の手法では、明確に評価値が確定するような局面は評価値分布がある値に収束するものとして表現され、モンテカルロ木探索を行う過程において自然にその分布が考慮されるため、よりシンプルにこの問題点を解決できると考えられる。

2.2 将棋に対するモンテカルロ木探索

橋本ら[2]、佐藤ら[3] など、将棋に対してモンテカル

⋆1 保木、山下らとの個人的情報交換による

ロ木探索の適用を試みた研究が存在する。純粋にプレイアウトを用いるこれらの研究は、従来のゲーム木探索より良好な性能は得られていない。

近年の機械学習を用いた手法の成功などにより、将棋は精度の高い評価関数を利用可能になっている。竹内ら[10] は、この評価値を利用し、ルート局面とプレイアウト末端の静止探索後の評価値の差を用いてプレイアウトの勝敗を定義する手法を提案した。問題集による評価では評価値を用いる効果が確認できたが、レーティングを用いた評価によると、従来のゲーム木探索に匹敵する性能はまだ得られていない。

我々は、将棋において十分成果を上げている、評価関数と min-max 木探索の双方の利点を活用することを目指す。

2.3 合　　議

将棋においては、近年「合議」という手法が提案された[5]。合議による探索では、探索の評価関数に異なる乱数を加えて複数回の探索を行い、得られた複数の探索木に対して、最も多く最善と選ばれた手を指す (多数決合議)、あるいは複数の探索木それぞれで得られる最善手の評価値のうち最良の評価値を持つ手を指す (楽観合議)、という方法で手を選択する。シンプルな方法であるが、性能は向上することが知られており、従来の逐次探索からの変更が少ないこと、並列化方法が自明で容易なことが利点である。ただし、投入したリソース量に対してどの程度の効果があるのか、どのような効率なのかに関してははっきりと議論されてはいない。

我々は、合議と同様に乱数を加えて複数の探索木を生成し、これをプレイアウトとして扱うモンテカルロ木探索を提案する。合議方式の結果から、評価関数に乱数を加えることで min-max 木探索の結果の信頼度がより高まることが期待され、その利点を単純合議以上に活用することを目指すものである。

なお、合議手法は、提案するモンテカルロ探索木を用いた探索手法では、初期局面においてのみプレイアウトを行い、木の展開を行わない手法、ととらえることもできる。

3. 提 案 手 法

モンテカルロ木探索アルゴリズムの基本的な構造は以下のようになる。

(1) 現在の探索木中のリーフ局面から何らかの方策でより正確に読みたいものを選ぶ。
(2) そのリーフが十分探索されていないならば、その局面からランダムに着手を行い、得られた最終局面での勝ち負けを判定する (プレイアウト)
(3) そのリーフからのプレイアウト回数が十分多いときには、そのリーフを展開し、探索木を成長させる
(4) その局面からの複数回のプレイアウトにおける勝率を探索木のリーフ局面の評価値とする。ルートに向かって評価値を伝搬させ、探索木の値を求める。
(5) 最終的に、得られた探索木の初手 (ルートの子供) の評価値から指すべき手を選択する。

本研究では、2 については「乱数を付加した探索によるプレイアウト」、4 については「bayesian approach による評価値伝搬」の手法を、それぞれ用いることを提案する。

3.1 乱数を付加した探索によるプレイアウト

ランダムな指し手を生成するのではなく、評価関数に乱数を加えた探索を複数回行い、その探索木で得られる指し手と評価値をプレイアウト結果とする。複数の探索により、木探索のリーフでは複数個の評価値が得られる。これをそのリーフの「確率分布を持つ評価値」として扱うことにする。

乱数付加においては、合議方式と同様に、探索開始局面からのプレイアウト回数と評価局面とをキーとした疑似乱数を生成して利用する。将棋の探索空間は合流が多いため、探索中に同一局面に至ったときに同じ乱数値を加えた評価値が得られるようにするためである。

3.2 bayesian approach による評価値伝搬

bayesian approach[11] は、リーフの値に確率分布があるときに、その確率分布を考慮したミニマックス探索を実現する手法である。チェスなどでは探索に関する既存の技法が利用できないことから有効性が低いとされてきた[12]。また、囲碁を模した単純なシミュレーションでは有望な結果が得られていた[13]。

本提案手法では、モンテカルロ木探索の評価値伝搬においてこの bayesian approach を利用し、リーフの評価値分布を考慮した探索木を構築する。評価値分布の伝搬を効率よく計算するために、リーフの評価値分布は離散的な確率分布であるとして複数のピンで表されるものとする。プレイアウトが 1 回の時には、評価値に δ だけずれを加えた値にも小さな確率を与えた擬似的な分布を生成し、bayesian approach における評価値分布とする。本論文の評価では、得られた評価値 v に 0.8、$v \pm \delta$ にそれぞれ 0.1 の確率を持たせた。またプレイアウト数が 2 以上の時は、$v \pm \delta$ の点の擬似的な確率は削除し、それぞれのプレイアウトで得られ

た評価値が、出現回数に従った確率で得られる確率分布であるとした。評価においては $\delta = 100$ と設定している。

なお、将棋の場合、探索を通して局面の勝ち負けが確定していることを判定することが可能である。これを反映するため、プレイアウトの結果、詰みだと判断され勝ち負けが確定した場合には、評価値が誤差を持たない、1 本のピンで表現される分布になるとした。

3.3 探索木の展開制御

リーフノードでは木全体で共通の最大試行回数 (プレイアウト数) が設定されており、プレイアウト回数がその閾値に達した場合に展開を行う。

探索木の展開制御について、複数の方法が考えられる。本研究では、以下の 3 つの手法を実装し、評価を行った。

3.3.1 幅優先探索制御

幅優先による順序でリーフを選択し、木を生長させていく手法である。初期局面ではすべての合法手を子として生成し、それ以外の局面ではミニマックス探索で有望な順に b 個のノードを選択して成長させる。全体としては分枝幅 b の探索木が生成されることになる。

なお、従来手法の一つである、金子らによる分散木探索手法[14] は、この幅優先探索制御に近い制御構造を持つ。金子らは、ルートに近い部分の探索木を分枝数を限定して静的に生成し、そこから探索を行うことを提案している。リーフの局面は 1 つの評価値を返す従来の木探索であり、本手法とは異なりプレイアウトは用いていない。

3.3.2 UCB による探索制御

UCB 値[1] は近年囲碁のモンテカルロゲーム木探索 (UCT) の探索制御に用いられ、高い性能が得られている。UCT においては、プレイアウトで得られるのはあるノードからゲーム進行した場合の勝敗数 (勝率と考えても良い) であり、本提案のようにプレイアウトで評価値分布が得られているときに適用するとどのような効果が得られるのかはまだわかっていない。

提案手法では、各中間ノードにおいて UCB 値が最大となるノードを選択し、そのノードがリーフノードならモンテカルロ試行 (プレイアウト) もしくは展開、中間ノードなら UCB 値による子ノード選択を再帰的に繰り返す、という処理を行う。

あるノード n における UCB 値は

$$UCB(n) = V_n + C\sqrt{\frac{\log T_{parent}}{T_n}}$$

として算出される。V_n はそのノードでの評価値の期待値であり、リーフノードではプレイアウトから得られる期待値、中間ノードではそのノードで得られている評価値分布に対する期待値となる。T_n はノード n での試行数、T_{parent} は n の親における試行数となる。試行数はリーフノードではプレイアウト数、中間ノードではそのノードの子孫に含まれる総プレイアウト数としている。C は試行で得られる評価値のぶれの程度を表現するスケールファクタである。C が大きくなると、不確定性をより大きなものと考え、評価値のぶれがより大きい、すなわち現在の試行で良い評価値を得られていないノードでも再度試す価値があると考えるようになる。よって、C が大きくなるほど全幅探索に近くなり、より網羅的に探索を行おうとするため、勝率は上がるが消費時間も増大すると考えられる。

3.3.3 QSS による探索制御

Baum らの Bayesian Approach[11] では、現在想定している最善手を上回る手があるときの確率重み付け変化量 Q^1:

$$Q^1 = \int_0^\infty P^1(q)qdq$$

(ここで、$P^1(q)$ は「真の最善手が現時点で想定される最善手 (1) を q だけ上回る確率」である) に対し、各リーフが与える変化量を “Q Step Size” と定義し、これを用いて探索制御を行うことを提案している。本提案では、この QSS がもっとも大きい、すなわちルートの確率分布と最善手の選択に与える影響が最も大きいと考えられるリーフから順に選択し、プレイアウトと展開を行うという手法をとった。ただし、探索終了条件を他の手法となるべくそろえるため、探索木の深さに制限を加え、PV を与えるリーフノードの深さが閾値に達したら探索を打ち切るという方法をとった。

4. 評　　価

4.1 実 験 環 境

3 通りの探索制御方式を用いたモンテカルロ探索を用いた提案手法と、従来技術である合議手法とを実装し、オリジナルのプレイヤとの対局を多数回行って手法の有効性を評価した。実装においては、コンピュータ将棋プレイヤ「激指[⋆1]」を利用し、その評価関数に模擬正規分布乱数を加えてプレイアウトを作成した。激指は実現確率打ち切り探索など既存の探索技法を多数導入した実用的なプログラムであり、世界コンピュータ将棋選手権などで優秀な成績を収めている。

テストは、実戦棋譜に出現した局面からランダムに局面を選んだ初期局面セットを作成し、その局面から先後を入れ替えて 1 回ずつ対局を行った。実験に用い

⋆1 http://www.logos.t.u-tokyo.ac.jp/~gekisashi/

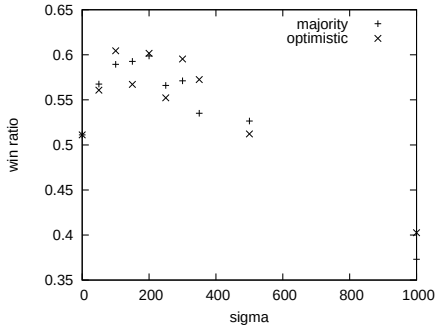

図 1 合議方式の勝率: σ による変化

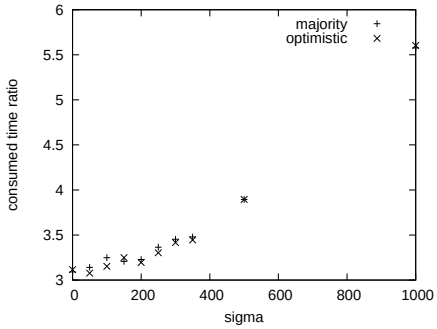

図 2 合議方式での消費時間 (比率): σ による変化

た初期局面数は 494 個、対局数はその 2 倍で 988 対局となる。

4.2 合議方式の性能

合議方式とオリジナルの激指との強さ比較を試みた。合議方式として、多数決を行うもの (majority) と、楽観的に、最良の評価値を返した手を選択するもの (optimistic) の 2 つを比較した。いずれの場合も合議方式のクライアント数は 3、複数のクライアントは逐次に別々の乱数で探索を行っている。

まず、合議において、評価関数に加える模擬正規分布乱数の σ をパラメタとし、勝率の変化を測定した。結果を**図 1** に示す。縦軸が勝率であり、0.5 がオリジナルと互角であったことを示している。多数決合議と楽観合議は、勝率にはそれほど大きな差はない。条件を変化させたとき、多数決の方が若干安定しており、楽観は不安定に勝率変化しているように思われる。また、得られる勝率は楽観の方が若干高い。

σ が 300 以下程度までは 0.5 を安定して越えており、最良時には 0.6 程度の勝率を得られている。σ が 300 を越えたあたりから勝率が落ち、あまり大きな乱数を加えると 0.5 を下回り弱くなることが見て取れる。なお、$\sigma = 0$ の時は合議版の評価関数に加えられる乱数値が 0 となり、オリジナルと同じ評価値を返すものになっている。この場合でも勝率が 0.5 を若干越えている理由は、合議クライアントが詰み探索結果を保持する Transposition Table を共有しているため、同一局面で複数回の探索を行うときに以前の詰み探索の結果を反映したより情報の多い探索が行われて、結果としてより正確な読みが行われたためであると考えられる。このような効果は、合議クライアントがメモリを共有しない並列実行時には得られないため、勝率向上の度合いを議論する時には割り引いて考える必要がある。とはいえ、詰み Transposition Table 共有の効果はそれほど大きなものではない。

また、オリジナルと合議方式が対局全体で消費した時間の比を**図 2** に示す。合議方式は完全に逐次に複数回の探索を行っているので、およそ 3 倍程度の時間が必要となることは予想される。図 2 に示したように、σ が 200 以下のあたりでは消費時間は予想と比較してそれほど変化していない。しかし、σ が 200 を超えたあたりから次第に消費時間が増加していることが見て取れる。これは、特に勝率が落ちている σ の大きい領域では、対局で自分が不利な局面に追い込まれることにより探索時間が増えたためであると考えられる。将棋のゲーム木探索においては、不利な局面では枝刈りの効率が落ち探索時間が増えがちであることが知られている。勝率が比較的高い $\sigma = 300$ 近傍での消費時間が増大している点の理由ははっきりしない。評価関数に加えられる誤差が大きくなるため、複数回の探索で得られる探索木のばらつきが大きくなっていることは確かであると思われる。

次に、合議でのクライアント数を変化させたときの評価を行った。評価値に加える乱数の分布 σ は 200 と設定した。クライアント数を 3,5,7,9,11,15 と変化させた時のオリジナルに対する勝率変化を**図 3** に示す。楽観合議の場合、クライアント数を 7 程度に増加させると勝率が向上しているが、それ以上クライアント数を増やしても勝率向上には結びついていない。勝率の推移が不安定なため明確な傾向があるとは言い切れないが、リソースを増やしてもある程度以上の勝率は得ることが難しいように推測される。また、多数決合議の方はおおむねクライアント数に関係なく一定程度の勝率で推移しており、やはり使用リソースを増加させてもこれ以上の勝率が得られるようには見えない。

なお、消費時間については想定通り、利用するクライアント数に比例した時間が必要となっていた (図は割愛する)。

4.3 bayesian approach に基づく探索手法の性能

bayesian approach による評価値分布を考慮した探索木の生成手法の性能を評価した。ここでは、幅優先探

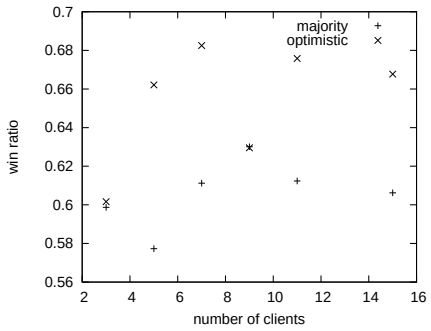

図 3 合議方式の勝率: クライアント数による変化

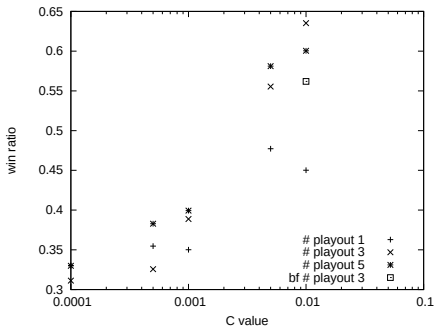

図 4 幅優先及び UCB による制御での勝率

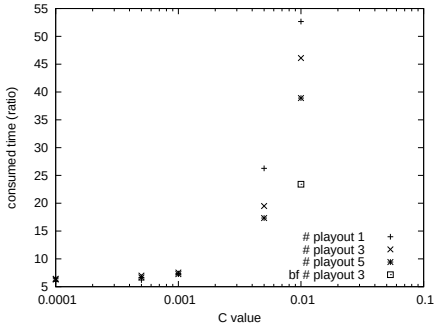

図 5 幅優先及び UCB による制御での消費時間 (比率)

索、UCB、QSS による探索制御を行うプレイヤを用い、オリジナルとの対戦性能評価を行った。幅優先、UCB どちらの場合も、評価関数に加える乱数は $\sigma = 200$ と設定した。また、プレイアウト数 1 の時の擬似的な確率分布は $\delta = 100$ とした。

幅優先と UCB による探索制御

幅優先探索において、探索木の幅は 3、すなわち、各ノードにおいてより深く展開する子ノードの数を 3 とした。また、リーフノードでのプレイアウト数は 3 である。UCB については、試行で得られる評価値のぶれの程度を表現するパラメタ C を変化させて実験を行った。C が大きいほど全幅に近い性質になると考えられる。

双方について、勝率の変化を**図 4** に示す。幅優先探索での結果は “bf # playout 3” の系列で示された点である。この結果の x 軸の値は意味がない。また、UCB について、あるノードを展開するまでの最大プレイアウト数を 1,3,5 と変えたものをプロットしている。なお、UCB 値の計算においては評価値を $[0,1]$ の範囲に線形に正規化した状態で計算している。C の値はこの範囲での調整を行っていることに留意されたい。

UCB において、プレイアウト数が 1 の場合は、どのような C の設定でもオリジナルの性能に達していない。均質な確率分布ではオリジナルの探索で得られる探索木と大きく異なる木が得られるわけではなく、浅い探索を繰り返して木を生成する提案手法ではオリジナルの読みの正確さが得られていないためであると考えられる。

プレイアウト数が 3 の場合は、$C \geq 0.005$ の領域ではオリジナルの強さを上回り、C を大きくしていくと高い勝率が得られるようになっている。プレイアウト数 5 の場合も同様であるが、プレイアウト数が増えると勝率の変化が若干小さくなるようにも見受けられる。

幅優先探索においても勝率は 0.5 を上回り、オリジナルよりは強いプレイヤが得られている。強さはおおよそ同じプレイアウト数の UCB 探索で $C = 0.005$ に設定した場合程度である。

また、消費時間の比を**図 5** に示す。UCB において、$C \leq 0.001$ の領域ではプレイアウト数の設定に関係なくそれほど大きなコストがかかっていない。しかし、オリジナルより強い $C \geq 0.005$ の領域においては消費時間が著しく増大している。ここで、プレイアウト数を大きくしていくと次第に消費時間が短くなっている点は興味深い。これは、プレイアウトを繰り返すことではなく、探索木の形が全幅に近くなっていることが消費時間の増大の主たる要因であることを示している。

幅優先探索においては、UCB で $C = 0.005$ と設定した場合と同程度の消費時間となっている。

QSS による探索制御

QSS を用いる探索制御において、ルート局面の期待値変化 (U_{all}) がどの程度小さくなるまで QSS による選択を続けるかを変化させる実験を行った。U_{all} が小さいほど、ルート局面で最善手として選択する手が誤っている可能性を小さくしようとするため、木探索がより正確になることが期待される。対戦は PV の深さをオリジナルに一致させた状態で行った。勝率の変化を**図 6** に示す。勝率は 0.5 以下であり、オリジナルには及ばない性能となっている。また、このときの消費時間比率を**図 7** に示す。U_{all} を小さくするほどプレ

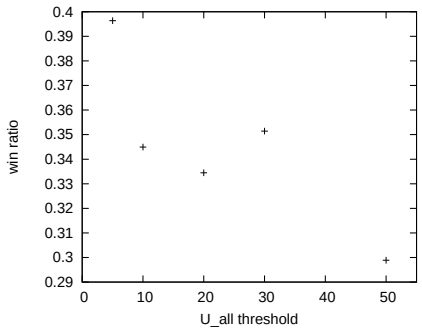

図 6 QSS による制御での勝率

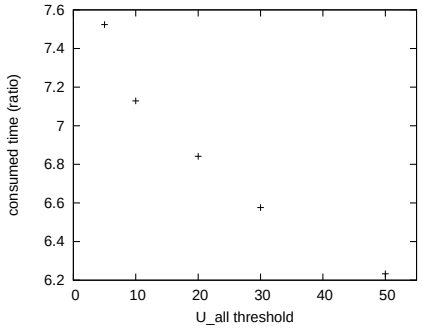

図 7 QSS による制御での消費時間 (比率)

イアウト回数が増え、消費時間が延びていることがわかる。

4.4 各手法の比較

各手法において、消費時間と得られる勝率の関係を**図 8** に示す。なお、全ての手法は逐次探索で実装されている。合議方式は多数決 (majority)、楽観合議 (optimistic) の 2 つについて、合議数を 3 から 15 まで変化させたときを示している。どちらの手法も、使用リソースを増加させていったときに勝率が上昇せず、0.6 から 0.65 程度で頭打ちになっている。UCB の場合、リーフでのプレイアウト数を 1 から 11 まで変化させたときの点が示してある。プレイアウト数を多くしていくと、次第に消費時間も少なく勝率も高くなってくるが、これはプレイヤが有利な局面にあるの時の方が消費時間が短くなる傾向にあることと、UCB による選択が効率的になることの双方に起因すると考えられる。また、プレイアウト数 5 以上の時は勝率向上もほぼ止まり、消費時間も延びてくる。なお、UCB 方式は単純幅優先 (breadth first の系列) よりは少ない消費時間で高い勝率が得られている。QSS については、PV の深さを 12 とオリジナルに一致させた場合 (QSS 12vs12)、root の期待値変化がどの程度小さくなるまで QSS を利用するかで設定を変更した時の結果を示しているが、どの設定でも勝率はオリジナルに劣る結果と

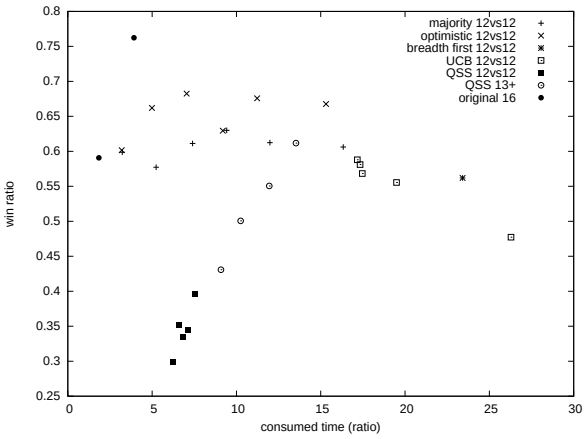

図 8 各手法における消費時間と勝率

なっている。ただし、消費時間は UCB、幅優先と比べて小さく保たれており、消費時間を増加させると勝率は向上していることも見て取れる。さらに、PV の深さ制限を 13 以上に深くしていった場合 (QSS 13+) では、オリジナルに勝ち越せるようになっており、消費時間も比較的小さい。また、利用リソース量の増加に伴って性能が単調に伸びており、今後の進展が期待できる結果であるといえる。

なお、参考としてオリジナルのプレイヤ (深さ 16) に対して、読みの深さを 15,14 と変化させて対局し、そのときの消費時間と勝率を測定したものを original 16 の系列に示す。original 13 は長時間使う方のプレイヤがおおむね深さ 13 までの場合 (対局した深さのペアが (12-10), (12-11), (13-11), (13-12) の 4 通り)、original 16 は深さ 16 までの場合 ((16-15), (16-14) の 2 通り) である。深さが 1 異なるときにオリジナルのプレイヤは 2.5 倍程度の時間を消費し、0.75 を越える高い勝率を得られる場合があることがわかる。オリジナルの思考方式を保ったまま処理を高速化した時の性能向上の目安であるといえるが、全く同じ評価関数、探索方式を用いての対戦実験は勝率の差が大きく見えがちになることも考慮する必要がある。また、深さ 13 程度では消費時間を増やしたときに大きく強さが変化しているが、深さ 16 に深くした場合では、強さの変化が小さくなっていることも読み取れる。つまり、浅い探索では一手深くすることの効果が大きいが、深い探索になるにつれその効果が小さくなっており、高速化による勝率への影響も小さくなっていることがわかる。

なお、オリジナルの激指において共有メモリ上で並列化を行ったときは、並列数を増やしていった場合でも 7 倍程度の速度向上しか得られておらず、より大規模な並列化の時には何らかの新たな探索手法を模索す

ることが必要になると考えられる[15]。

合議法では、分散計算への適用は容易だが、リソースを増加させても勝率向上への効果が見えておらず、得られる強さには限界が見えている。また、合議法では、並列実行可能な箇所が初期局面のプレイアウトの並列実行のみに限定されており、大規模並列環境への適用も何らかの工夫が必要になると考えられる。

bayesian approach を用いた方法では、勝率の性能向上が実現されており、分散計算への適用もゲーム木探索アルゴリズムの直接的な分散化と比較すると容易であると考えられる。本実験での勝率向上は合議法に届かないものとなったが、提案手法の方がクリティカルパスが短く、分散計算による高速化の余地が合議法より高いため、将来の性能向上が期待できると考えている。

モンテカルロ木探索を並列化する際、ルートが同じ複数のモンテカルロ木探索を独立して実行し、最終的にルート局面で複数のプレイアウト結果を単純に足し算するという手法[16] が利用されることが多い。本提案手法で同様の並列化が可能なのか、どの程度有効なのか、など、他の並列化手法の検討も含めて今後の課題としたい。

5. 終わりに

本研究では、従来用いられてこなかったモンテカルロ探索を将棋に適用するため、

- 評価関数に乱数を与えることでモンテカルロ探索のプレイアウトを構成する
- bayesian approach を用いることで評価値分布を反映したモンテカルロゲーム木を構築する

という手法を組み合わせるという探索アルゴリズムを提案し、その有効性を評価した。

提案手法は使用リソースの増加に伴って勝率を向上させることができ、分散計算に向いた新しい探索手法として有望であることが明らかになった。従来の合議手法との比較を行った際、使用リソースと得られる勝率の比では従来手法に及ばないものの、分散計算での性能向上の余地は提案手法の方が高く、有効な分散計算アルゴリズムを構築することが望まれることが判明した。

今後、分散計算への適用のために、具体的な分散アルゴリズムの検討、実装、性能向上並びに探索制御部分のオーバヘッドなどの評価を行っていく。

参 考 文 献

1) Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. Technical Report RR-6062, INRIA, 2006.
2) 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, 2006.
3) 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, 2008.
4) Mark H.M. Winands and Yngvi Björnsson. Evaluation function based monte-carlo LOA. *ACG*, pp. 33–44, 2009.
5) 伊藤毅志, 小幡拓弥, 杉山卓弥, 保木邦仁. 将棋における合議アルゴリズム — 多数決による手の選択. *IPSJ*, Vol.52, No.11, pp. 3030–3037, Nov 2011.
6) Rémi Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *CG 2006*, 2006.
7) Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the 17th European conference on Machine Learning*, ECML'06, pp. 282–293, 2006.
8) RichardJ. Lorentz. Amazons discover monte-carlo. In *CG 2008*, pp. 13–24, 2008.
9) Mark H. Winands, Yngvi Björnsson, and Jahn-Takeshi Saito. Monte-carlo tree search solver. In *CG 2008*, pp. 25–36, 2008.
10) 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.
11) EricB. Baum and WarrenD. Smith. A bayesian approach to relevance in game playing. *Artificial Intelligence*, Vol.97, No. 1–2, pp. 195–242, 1997.
12) A. Junghanns. Are there practical alternatives to alpha-beta in computer chess? *ICGA Journal*, Vol.21, No.1, pp. 14–32, 1998.
13) Gerald Tesauro, V. T. Rajan, and Richard Segal. Bayesian inference in monte-carlo tree search. In *UAI*, pp. 580–588, 2010.
14) 金子知適, 田中哲朗. 最善手の予測に基づくゲーム木探索の分散並列実行. 第 15 回ゲームプログラミングワークショップ, pp. 126–133, 2010.
15) 横山大作.「激指」におけるゲーム木探索並列化手法. 人工知能学会誌, Vol.26, No.6, pp. 648–654, Nov 2011.
16) Guillaume M.J.-B. Chaslot, Mark H.M. Winands, and H.Jaap vanden Herik. Parallel monte-carlo tree search. In *CG 2008*, Vol. 5131 of *LNCS*, pp. 60–71, 2008.