ベイジアンアプローチに基づく 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,a)]

概要：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、Bayesian Approach による評価値伝搬を利用したモンテカルロ木探索を提案し、将棋への適用を試みている。対戦による評価の結果、提案手法の有効性が確認されるとともに、シミュレーション探索をある程度以上大きくする必要があるなど、提案手法の特性に関する知見が得られた。また、シミュレーション回数の増加が有効でないなどの問題点も明らかになった。

An Application of Monte-Carlo Tree Search Algorithm for Shogi Player Based on Bayesian Approach

Daisaku Yokoyama[1,a)]

Abstract: 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 uses Bayesian Approach to propagate distributions of leaf values, and apply it for Shogi player. Through large amount of self-play evaluations we conclude the method has high effectiveness. It also reveals several characteristics of the proposed method; simulation search should keep a certain amount of size, increasing the number of simulations is not effective, etc.

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

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

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]。

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。筆者らは、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した研究を行っている [5]。

本稿では、Bayesian Approach に基づくモンテカルロ木探索を利用した将棋プレイヤの作成方法について、その探索アルゴリズムの詳細を示すとともに、複数のパラメタについて性能に関する影響を詳細に調査し、アルゴリズムの性質の理解と有効性の検証を試みる。

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、シミュレーション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。

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 [11]。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。

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シミュレーションを行い、木の展開を行わない手法、ととらえることもできる。

2.4 Bayesian Approach に基づく探索手法

筆者らは先行研究 [5] で、Bayesian Approach に基づくモンテカルロ木探索手法を提案し、幅優先、UCB 値、QSS のそれぞれを用いた探索制御手法を用いて提案手法の有用性を評価するとともに、関連が深い既存手法の合議による探索も実装し、提案手法との性能比較を試みた。オリジナルプレイヤとの多数の対局実験を通して、合議法は有効であるがリソースを増やしたときの性能向上に限界が見られること、Bayesian Approach に基づく手法では、合議には性能が及ばないが有効性は確認され、QSS による制御手法がリソース増加時に期待が持てるという結論が得られた。ただし、この研究では提案手法の性能を左右する種々のパラメタ設定に関する検証が不十分であった。

3. 提案手法

筆者らは、先行研究 [5] において、「乱数を付加した探索によるシミュレーション」、「Bayesian Approach による評価値伝搬」の 2 つの手法を利用したモンテカルロ木探索手法を提案した。本稿では、この提案手法について、探索制御のアルゴリズムなどを詳細に説明する。

3.1 乱数を付加した探索によるシミュレーション

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シミュレーション結果とする。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シミュレーション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。

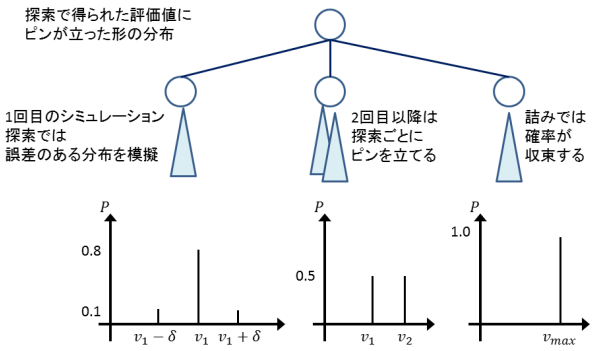

図 1 シミュレーション結果による評価値分布の作成手法

3.2 Bayesian Approach による評価値伝搬

Bayesian Approach [12] 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 [13]。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 [14]。

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。

図 1 はシミュレーションの回数とそこで得られる評価値分布を示している。シミュレーションが 1 回の時には、得られた評価値 (v_1) 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する。またシミュレーション数が 2 以上の時は、$v_1 \pm \delta$ の点の擬似的な確率は削除し、それぞれのシミュレーションで得られた評価値 (v_1、v_2 など) が、出現回数に従った確率で得られる確率分布であるとする。

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、シミュレーションの結果、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

提案手法は、モンテカルロ木探索アルゴリズムを基礎としており、リーフノードを選択してシミュレーションを行い、シミュレーション回数が設定された一定回数 ($Simnum$) 以上に達した場合にはそのノードを展開して木を成長させる、という動きが基本となる。探索木の展開順序の制御、終了判定などのアルゴリズムを**図 2** に示す。また、探索制御に用いているパラメタの説明を**図 3** に示す。$Playdepth$ はプレイヤの強さをおおむね規定することを期待した深さパラメタであり、提案手法ではおおむね $Playdepth$ の長さ

[探索のメインループ]
while *root* の終了条件が満たされない:
  $refine\_p$ = **find_refine()**
  $refine\_p$ に関してシミュレーション or 展開
  for p **in** [$refine\_p$ から *root* まで上る]:
    p の確率分布をアップデート
    p が詰まされているなら再チェック
  root で U_{all} を求める
  QSS のための ESS を *root* から leaf まで再計算

[展開ノードの選択]
function find_refine():
  if U_{all} が一定回数 (100 回) 以上変化していない:
    return PV の先頭
  if U_{all} が閾値 ($Uall\_th$) 以下:
    return PV の先頭
  $leaves \leftarrow$ 末端ノード
  $leaves$ を ESS の大きいものから降順にソート
  $leaves$ を先頭 1/10 のみ残す
  for p **in** [$leaves$]:
    if p の深さ $+Simdepth < Playdepth$:
      return p
  // $leaves$ の simulation が全て $Playdepth$ に達した
  return PV の先頭

[展開処理]
root の場合は全幅、それ以外では $max(12-depth \times 2, 3)$ に従う数の子ノードを展開する

[終了条件]
- *root* の確率分布が収束したら終了 (詰み・詰まされの場合)
- PV の先頭ノードが $Simnum$ 以上のシミュレーションを行い、$depth(PV)+Simdepth >= Playdepth+PVth$ ならば終了

図 2 探索制御アルゴリズム

だけの PV が最終的に得られることを期待した制御を行っている。$PVth$ は、得たい PV 長をどの程度延長するかを定めるパラメタとなる。

本論文で検証を試みている手法は、Bayesian Approach で提唱されている、ルートノードの確率分布に対して最も影響を与える度合いが大きいリーフノードを選択して展開する、という QSS(Q Step Size) による探索制御アルゴリズムである。Bayesian Approach では、ルート局面の期待値変化の見積もり (U_{all}) が一定以上小さくなったところで探索を打ち切るという終了条件を用いていたが、PV 長が $Playdepth$ 程度出力されることが望ましいと考えたため、U_{all} が小さいときは現在の PV ノードを展開する、という手法をとっている。また、リーフノードの 1/10 が $Playdepth$ に到達した場合も、それ以上 QSS による最良

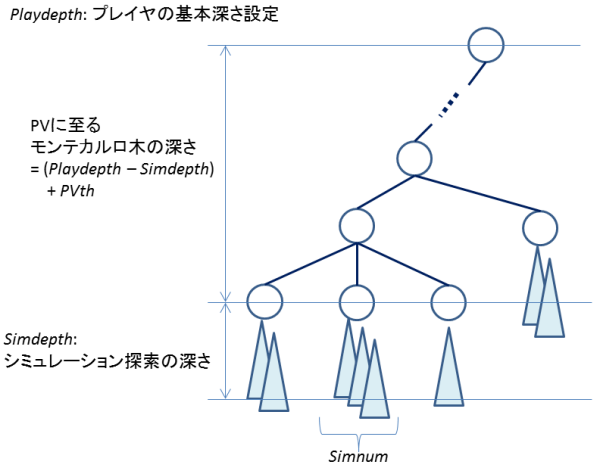

図 3 提案手法における探索制御パラメタ

優先探索を続けず、現在の PV ノードを展開するようにしている。その他、展開幅の制御などを含め、ある程度現実的な探索時間で結果を返すためのアドホックな制御を加えているが、これらについて詳細な検討は行っていない。

3.4 性能を左右する設計項目

本手法の探索戦略においては、大きく以下の項目に性能を左右する設計項目、及びトレードオフ点が存在すると考えられる。

シミュレーションに加える乱数分布 加える乱数が小さい場合はシミュレーション結果が変化せず、シミュレーションとしての働きが弱くなるが、乱数を大きくすると探索の精度に影響が及び、妥当なシミュレーション結果が得られなくなるため、何らかのトレードオフ点があると考えられる。

シミュレーション結果を用いた評価値分布の作成手法 Bayesian Approach では、リーフに評価値の確率分布があるミニマックス木を容易に扱うために、確率分布をいくつかのピンによって表現する。そのため、シミュレーションを複数回行った結果からこの確率分布を作成する何らかのモデルが必要になる。特に、リーフにおけるシミュレーションの回数が少ないときには、評価値の信頼度が低いことを適切に表現するような手法が必要となる。本稿の手法では、図 1 のように、1 回めのシミュレーション結果について得られた評価値 v に対し $v \pm \delta$ の点にピンが存在する確率分布を作成し、誤差の存在を表現している。

シミュレーション探索部分の大きさ ($Simdepth$) 1 回のシミュレーションに利用するリソース量を多くすると、シミュレーションの精度は向上するが、逐次部分の割合が増えるため並列化効率が低下する可能性がある。

本手法において、シミュレーション部分を最大まで大

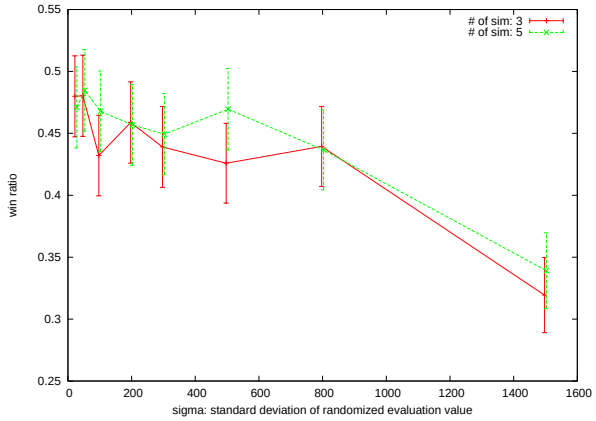

図 4　評価値に付加する乱数分布変更時の勝率

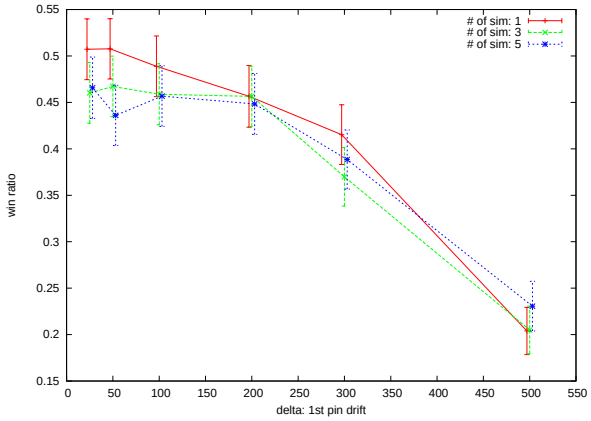

図 5　初期確率分布変更時の勝率

きくし、ルートノードでのみシミュレーションを行うようにすると、既存手法である合議 [11] による制御に極めて近いアルゴリズムとなる。

シミュレーション数閾値 (*Simnum*)　シミュレーションがある程度精度良く局面評価を行えると期待できるため、従来のシミュレーションよりは少ない回数でモンテカルロ木の末端評価が収束し、より少ないシミュレーション数で展開を行う方が効率がよい可能性がある。

本論文では、将棋を対象問題として本アルゴリズムの実装を行い、多数の対戦を行ってこれらの項目に対する評価を試みる。

4. 評価

4.1 実験環境

提案のモンテカルロ探索手法を実装し、オリジナルのプレイヤとの対局を多数回行うことで手法の性質と有効性を評価した。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シミュレーションを作成した。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。

テストでの勝率測定は、実戦棋譜の 31 手目の局面からランダムに 500 局面を選択し、その局面から先後を入れ替えて 1 回ずつ、合計 1000 対局によって求めた。オリジナルプレイヤの設定は得られる PV 長がおおむね 12 になるようなものとし、提案手法のプレイヤもそれに相当するよう *Playdepth* を 12 に設定した。それぞれの 1000 対局においては、おおむね 500〜1000 時間程度の CPU 時間を要した。

また、以後の実験結果全てにおいて、勝率については 95%信頼区間をエラーバーとして示してある。

4.2 乱数分布の影響

シミュレーション探索に付加する正規分布乱数の分布を変化させて評価を行った。*Simdepth* を 8、*PVth* を 4 に固定し、乱数の標準偏差 σ を変化させたときの勝率推移を**図 4** に示す。各系列は *Simnum* を 3,5 と設定した場合である。なお、グラフの見やすさのためにそれぞれの実験結果の点はわずかにずらしてプロットしてある。また、激指においては評価値 100 が歩一枚の駒価値に相当している。

σ が 800 以下程度の領域では、勝率はそれほど大きく変化していない。しかし、全般的には σ が大きくなると勝率が下がる傾向にあるように見受けられ、1500 程度まで大きくなるとはっきりと差が現れる。

以降の実験では、$\sigma = 200$ の乱数を用いている。

4.3 初期確率分布の影響

モンテカルロ木のリーフを展開し、シミュレーションを 1 回だけ行ったノードにおいては、誤差のある確率分布を模擬するため、シミュレーションで得られた評価値 v に対して δ だけ離れたところにピンを立てる (図 1)。この仮想的な確率分布の形の設定がどのような影響を与えるかを調べるため、δ を変化させて対局による評価を行った。

Simdepth を 800、*PVth* を 4 と固定し、δ を変化させたときの勝率変化を**図 5** に示す。各系列はノード展開に必要なシミュレーション回数の閾値 *Simnum* を 1,3,5 と変化させたときの結果である。横軸は δ であり、それぞれの系列で 25, 50, 100,200,300,500 と変化させているが、グラフでは見やすさのためにわずかに横方向にずらしてプロットしている。ただし、$\delta = 500$ の点のみ、$\pm\delta$ 地点の確率分布の値が 0.25 と異なって設定されているため、参考結果として掲載する。

今回の実験の範囲では、*Simnum* の設定によらず δ が小さい方が高い勝率を得られる結果が見て取れる。*Simnum* が 1 の場合、δ は 50 以下の範囲で勝率が高く、それ以上大きくした場合には勝率が低下する。一方、*Simnum* を 3 以上にした場合は、δ が 200 以下の範囲ではあまり明確な勝率変化は現れない。これは、*Simnum* が 1 より大きい場合には、シミュレーション探索の評価値に加えられる乱数

[*1] http://www.logos.t.u-tokyo.ac.jp/~gekisashi/

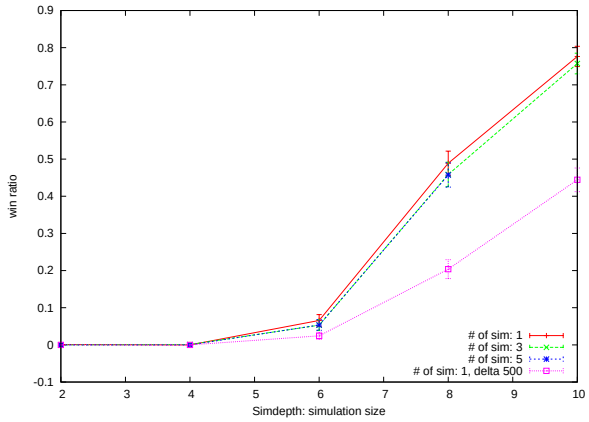

図 6 シミュレーション探索のサイズ変更時の勝率

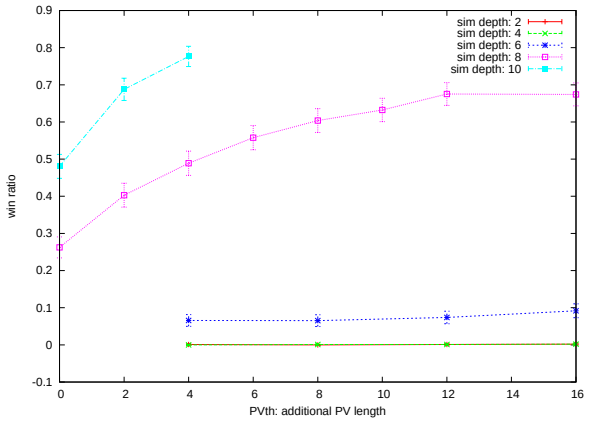

図 7 PV 長変更時の勝率

($\sigma = 200$) が最終的に得られるモンテカルロ木の性質を規定しており、そのシミュレーション結果の分布と比較して 1 回目のシミュレーションの確率分布範囲が小さい場合には、δ は最終的な結果へ影響を及ぼしにくい、という理由によるものと推察される。

また、*Simnum* を増やしても勝率にはそれほど影響が現れない、という結果も見て取れる。現在の手法について、モンテカルロ木が初期確率分布に強く影響を受けており、シミュレーションによってあまり修正されない、という状況にあると推察される。

4.4 シミュレーション単位の影響

δ を 100、*PVth* を 4 に設定し、シミュレーション探索のサイズ *Simdepth* を変化させたときの勝率変化を**図 6** に示す。グラフの系列は *Simnum* をそれぞれ 1,3,5 と設定した場合を示している。また、*Simnum* を 1、δ を 500 とした設定での結果を系列 “# of sim: 1, delta 500” に示している。

Simdepth が 6 より小さい場合は勝率がほぼゼロであり、ある程度以上の規模でシミュレーション探索を行わないと、オリジナルのアルファベータ探索の強さに達しないことがわかる。また、異なる δ でもこの傾向は同様であることが見て取れる。

また、*Simdepth* を 2 から 10 までの値で固定し、*PVth* を変化させたときの勝率変化を**図 7** に示す。*Simdepth* が 6 より小さい場合には、モンテカルロ木の深さを深くしていっても効果がないことがわかるが、*Simdepth* を 8 まで大きくすると、モンテカルロ木の深さを深くすることで勝率が向上し、提案手法で効果が得られることが確認できる。ただし、木の深さを深くしたときの勝率向上の度合いは漸減していき、深さ 12 程度で向上が頭打ちになることも見て取れる。*Simdepth* を 10 まで大きくすると勝率はさらに高くなり、シミュレーションサイズは大きいほど強くなるという結果が得られた。

図 8 はこの実験結果について、オリジナルプレイヤと比較した消費時間の比を横軸にしてプロットしたものである。*Simdepth* が 10 の場合は、8 の場合と比較して同一の *PVth* 設定での消費時間が 2.5〜3 倍程度延びており、同一の消費時間を要する領域で比較すると、*Simdepth* が 8 の方が高い勝率を得られる場合もあることがわかる。使用リソースに対する効率の良さと言う意味では必ずしもシミュレーションサイズを大きくした方が良いとは言い切れないと考えられる。また、シミュレーションサイズを大きくするということは、逐次実行部分の粒度を大きくするため、並列計算の適用を考える場合にはより不利になることが考えられる。

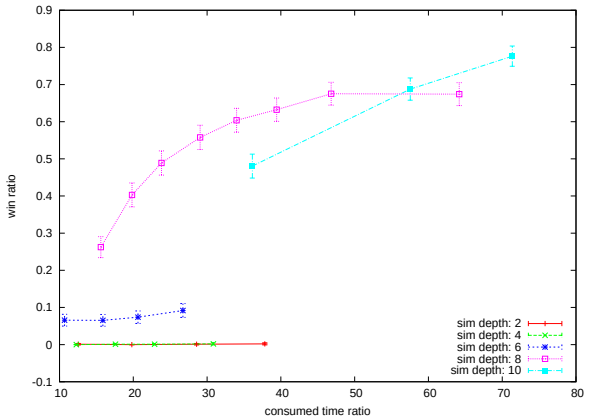

図 8 PV 長変更時の消費時間比率と勝率

4.5 展開に要するシミュレーション回数の影響

Simdepth を 8 に固定し、*Simnum* を 1 から 7 まで変化させたときの勝率変化を**図 9** に示す。それぞれの系列は *PVth* を変化させた場合を示している。どの系列においても、*Simnum* を増加させても勝率は向上しない結果が見て取れる。全般的に、*Simnum* を 1 から 3 に変化させたときに比較的大きく勝率が下がり、それ以降は *Simnum* を増加させてもほぼ横ばいの推移を行っているように見える。*Simnum* = 1 の時は、シミュレーション探索で得られた評価値には誤差は加えられておらず、実質的には激指の持

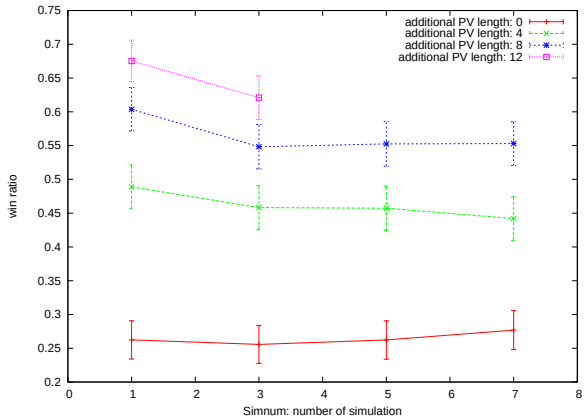

図 9　シミュレーション回数閾値と勝率の関係

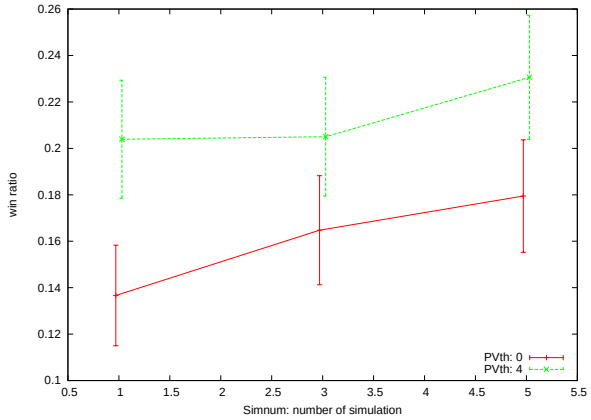

図 10　シミュレーション回数閾値と勝率の関係 ($\delta = 500$)

つ評価関数の値がそのまま用いられているが、シミュレーション回数を増やすと誤差の入った評価値が用いられるようになり、今回の実験結果にはその悪影響が現れたと考えられる。

参考までに、初期確率分布の δ を 500 とし、$v \pm \delta$ の点の確率を 0.25 としたプレイヤで、$Simnum$ を変化させたときの勝率変化を**図 10** に示す。これは、シミュレーション 1 回目に得られた評価値の確率分布がより不確かなものとして扱われるような設定となる。勝率が低い領域での結果であり、あまり明確な差が出ているわけではないが、シミュレーション回数を増やしたときに若干勝率が向上している傾向が見られる。シミュレーション回数が少ないときの分布の扱い方を変更することで、手法が改善される可能性を示唆しているといえる。

4.6　勝率と所要時間の関係

これまでの評価結果をもとに、代表的なパラメタ設定を用いて提案手法の勝率と所要時間の関係を調べた。

δ を 100，$Simdepth$ を 8 とし、$PVth$ を変化させたときの勝率を**図 11** に示す。各系列は $Simnum$ を 1 から 7 まで変えた場合である。$Simnum$ が 1、3 の系列では $PVth$ を 0 から 12 まで、それ以外の系列は $PVth$ を 0 から 8 ま

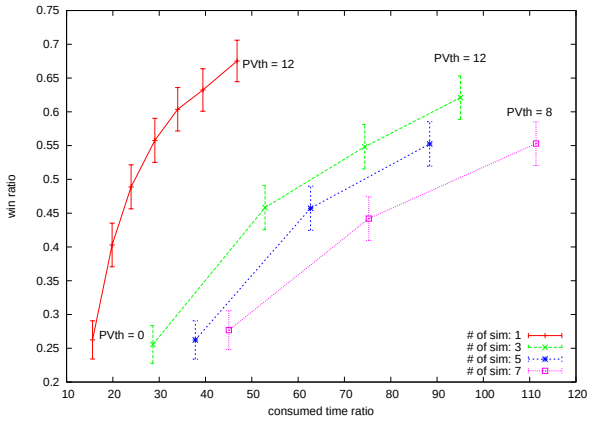

図 11　勝率と所要時間の関係

で変化させている。横軸は対局で消費した時間をオリジナルプレイヤとの比率で示している。

いずれの設定においても、$PVth$ を延ばしていくことで勝率は向上し、実験の範囲ではまだ限界には達していないように見受けられる。前述したように、$Simnum$ を増やしたときには勝率がむしろ低下しており、消費時間が伸びることも考慮すると、現在の手法で $Simnum$ を増やすことはあまり有効ではないと結論づけられる。

4.7　既研究との比較

今回の実験においては、時間切れ負けとする時間設定を変更し、より長い持ち時間を用いて対局を行っている。そのため、既発表 [5] の実験と比較して、提案手法の消費時間が長い結果が得られている。特に、一部の対局において提案手法が特異的に時間を要する場合が見受けられ、このような対局が所要時間を大きく引き上げている可能性がある。これは提案手法において、時間制御を工夫することの重要性を示唆しているが、本稿では、アルゴリズム本来の性能評価を行うことを目的としているため、時間切れ判定の他に制御を加えないで実験を行った。

4.8　考察

本実験を通して、モンテカルロ木の末端ノードでのシミュレーション回数が 1 回の時が最も有望な性能を出し、シミュレーション回数を 3 回以上に増やしていっても効果が得られないばかりか、逆に性能低下の傾向が見られた。ただし、これは乱数を利用した合議による手法が有効であるという既知の知見 [11][5] とはやや合致しない結果である。ノードの確率分布について、静的な評価値と固定的に定められた誤差から生成したものと、乱数付加したシミュレーション探索を繰り返して生成したものとで、真の分布をより精度良く再現しているのはどちらなのか、真の分布からのずれが Bayesian Approach による確率分布伝搬によって最終的にどの程度探索結果に反映されるのか、などを詳細に観察、検討する必要があると考えられる。また、

今回は全てのノードで同一の誤差を固定的に付加しているが、真の確率分布はノード毎に異なると考えられ、その違いを反映することで探索がより精度良く行えると思われる。これは、元々の Bayesian Approach の提案でも指摘されている [12]。Baum らは局面の特徴量をもとにクラスタリングを行って確率分布を変化させたが、乱数付加したシミュレーションを繰り返すことは、このような局面の確率分布の違いを自然に反映することにつながると考えられるため、真の確率分布がうまく推定できないような局面に限ってシミュレーションを行う、などの改良手法を考えることもできる。

また、初期確率分布が木の形をおおむね決めてしまい、その後のシミュレーションが木の形を修正するのに役立っていないように推察される結果も得られている (図 5)。偶然得られたシミュレーション結果の影響が修正されないのは望ましい動きが実現できているとは言えず、原因を詳細に探る必要があると考えている。シミュレーション回数が少ないときの確率分布の扱い (図 10) を変更して評価する、シミュレーション回数を大幅に増やしたときの振る舞いを観察する、などを今後検討したい。

一方で、シミュレーション回数 1 回のみでも、Bayesian Approach によってオリジナルより強いプレイヤが作成できたことは有用な知見であると考えられる。乱数を用いたシミュレーションを行わない場合でも、Bayesian Approach による最良優先探索を行っている部分は並列に値を求めたいノードが多数存在しており、並列探索の適用が容易である。一般的に想定されるモンテカルロ木探索の枠組みとは少し異なってくる可能性もあるが、確率分布を用いた最良優先探索と従来のアルファベータ探索を組み合わせた、並列化に適した探索手法の 1 つとして位置づけることもできると思われる。

今回の実験では、提案手法はオリジナルに対して極めて長い消費時間を要する結果となっているが、実験用のプレイヤは実装面で高速化の工夫をあまり施していないため、実際には消費時間比率はもう少し改善されると期待できる。もちろん、オリジナルプレイヤよりリソースを要することは確実であり、並列処理の適用しやすさでその不利をカバーすることを目指す手法であるため、並列処理を適用した時の振る舞いについては今後検証する必要がある。

5. 終わりに

本研究では、筆者らが提案してきた、

- 評価関数に乱数を与えた探索を用いてモンテカルロ探索のシミュレーションを構成する
- 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モンテカルロ木探索アルゴリズムについて、その構築方法の詳細を示すとともに、性能を左右するであろう設計パラメタについて評価実験を行い、提案手法の性質と有効性を理解することを目指した。

オリジナルプレイヤとの対局実験を通して、提案手法によって強さを向上させることができることを確認した。シミュレーションに用いる探索のサイズはある程度以上大きくなければ有効でないこと、適切な探索サイズは消費時間との関係で変わりうること、などの特性に関する理解が得られた。また、シミュレーション回数が少ないときの確率分布の扱いがモンテカルロ木の性質を大きく決定しており、シミュレーション回数を増やしてもその結果があまり反映されていないという問題点も明らかになった。シミュレーション回数が少ない状態でも有効な並列実行に適した最良優先探索としての発展、シミュレーション回数を増やしたときにも効果が得られるようなモンテカルロ木探索としての発展、の両面から提案手法を検討し、改善を図っていきたい。

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] 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. 第 17 回ゲームプログラミングワークショップ, pp. 76–83, 2012.

[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] Richard J. 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] 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.

[12] Eric B. Baum and Warren D. Smith. A bayesian approach to relevance in game playing. *Artificial Intelligence*, Vol. 97, No. 1–2, pp. 195–242, 1997.

[13] A. Junghanns. Are there practical alternatives to alpha-beta in computer chess? *ICGA Journal*, Vol. 21, No. 1, pp. 14–32, 1998.

[14] Gerald Tesauro, V. T. Rajan, and Richard Segal. Bayesian inference in monte-carlo tree search. In *UAI*, pp. 580–588, 2010.