探索窓の予測分布を用いた並列$\alpha\beta$探索

浦 晃[1,a)]　鶴岡 慶雅[1,b)]　近山 隆[1,c)]

概要：大規模環境を用いた並列 $\alpha\beta$ 探索の性能向上のためには、枝刈りされにくいノードを優先して探索することが重要である。我々は各ノードが枝刈りされない確率を推定し、この確率が大きいノードから順に最良優先探索する手法を提案している。本論文では、この確率を精度良く求めるために、探索窓の α と β を確率分布で予測し、それを用いて各ノードが枝刈りされない確率を計算する方法を提案する。将棋を用いた評価では有効性を示すことができなかったものの、人工木を用いた評価では分枝数が少ないほど提案手法の有効性が見られることが分かった。

Parallel $\alpha\beta$ Search with Estimated Distributions of Search Windows

Akira Ura[1,a)]　Yoshimasa Tsuruoka[1,b)]　Takashi Chikayama[1,c)]

Abstract: To achieve high performance in parallel $\alpha\beta$ search on a large-scale platform, it is important to prioritize those nodes that are unlikely to be pruned. In a previous paper, we have proposed a method to estimate probabilities of nodes' not being pruned and to perform best-first search using those probabilities as priorities. In this paper, to estimate those probabilities more accurately, we propose a method to estimate the distributions of search windows and to calculate the pruning probabilities using those distributions. The evaluation results with artificial game trees show that the proposed method becomes more effective as the branching factor is decreased; however, its effectiveness on real shogi game trees has not been observed.

1. はじめに

マルチコア化などによる利用可能なプロセッサのコア数の増加に伴い、大規模な計算環境を用いた、強いコンピュータゲームプレイヤを作るための並列 $\alpha\beta$ 探索への要求が高まっている。実際に、2012 年の世界コンピュータ将棋選手権[*1]では、GPS 将棋[*2]が 3224 コアを用いているうえ、10 コア以上を用いた参加チームも珍しくなくなってきている。

並列 $\alpha\beta$ 探索アルゴリズムの一つとして、Young Brothers Wait Concept (YBWC) [3] と呼ばれる手法があり、これを用いた並列 $\alpha\beta$ 探索が広く研究されている [4], [5], [7], [9]。YBWC は最も有望な子ノード (最左の子ノード) の探索の終了を待ってから残りの子ノードを並列に探索するという手法であるが、先行する探索の終了を待つために、大規模環境ではアイドル状態のプロセスが多くなり、台数効果が頭打ちになる [1], [11]。その問題を解決するため、アイドル状態のプロセスがある限り次々とタスクを割り当てる投機的実行 [11] を我々は提案している。さらに、投機的実行に合わせて、各タスクが探索終了まで枝刈りされない確率を推定し、この確率が大きいタスクから順に実行する手法も提案しているが、実探索による評価では有効性を示すことができていない [12]。そこで、調べたところ、確率の推定が正しくなかったことが分かった。そこで本研究では、確率を精度良く推定するために、探索窓の α と β を確率分布で予測し、それらを用いてノードが枝刈りされない確率を計算する方法を提案する。

2. 提案手法

我々は、並列 $\alpha\beta$ 探索において、各ノードが枝刈りされ

[1] 東京大学大学院工学系研究科
Graduate School of Engineering, The University of Tokyo
[a)] ura@logos.t.u-tokyo.ac.jp
[b)] tsuruoka@logos.t.u-tokyo.ac.jp
[c)] chikayama@logos.t.u-tokyo.ac.jp
[*1] http://www.computer-shogi.org/wcsc/
[*2] http://gps.tanaka.ecc.u-tokyo.ac.jp/gpsshogi/

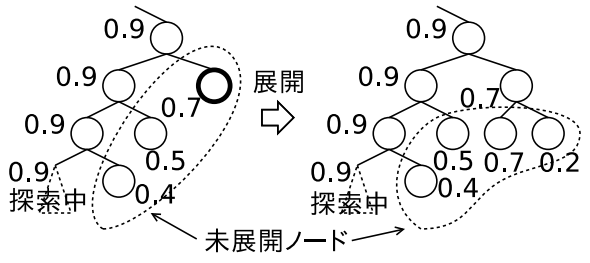

図 1 探索の進行例

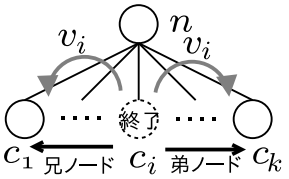

図 2 並列探索窓の更新方向

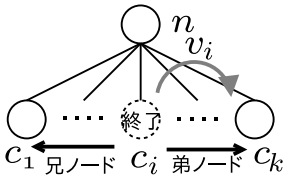

図 3 逐次探索窓の更新方向

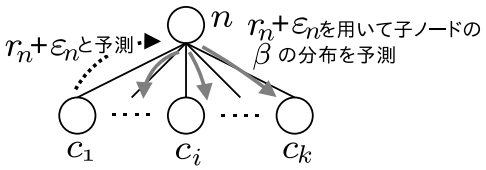

図 4 c_1 の予測を用いた β の分布の予測

ない確率を推定し、それを優先度に用いて最良優先探索する手法を提案している。つまり、図 1 に示すように、どのプロセスにも排他的に探索されていない未展開ノードのうち、最も枝刈りされなさそうなノードを展開し、子ノードをゲーム木に付け加えることによって探索を進めていく。

このときの探索の順序を決定する優先度、つまり、ノードが枝刈りされない確率を精度良く求めることが重要である。既に我々は、ノードが枝刈りされない確率として、そのノードが、PV ノード、ALL ノード、CUT ノードのそれぞれになる確率の和として計算する手法を提案している [12]。$\alpha\beta$ 探索では、全てのノードで子ノードの探索順序が正しいときでも探索しなければならないノードが存在し、それらのノードからなる木を最小探索木と呼ぶ。最小探索木に含まれるノードは、PV ノード、ALL ノード、CUT ノードの 3 つに分類される。つまり、あるノードがこれらのどれかとなる確率を求めれば、それがそのノードが枝刈りされない確率であるという発想である。しかし、実際に人工木を用いて繰り返し逐次探索を行い、そのときに枝刈りされた回数を統計的に調べ、探索開始直後にこの手法を用いて推定した確率と比較したところ、ごく小さな探索木を用いた場合ですら確率の推定が正しくなかったことが分かった。

そこで本研究では、ノードが枝刈りされない確率を精度良く求めることを目的とする。そのために、探索窓を確率分布で予測し、そこから枝刈りされない確率を計算する手法を提案する。

2.1 探索窓の推定

本研究では探索窓の α と β を確率分布を用いて予測することを提案する。探索窓は他のノードの評価値によって決まるため、探索窓を予測するためには他のノードの評価値を予測しなければならない。本論文では、ノードを探索して得られる評価値は、それより浅い探索で得られる評価値の周りに正規分布すると仮定する。さらに、この仮定を多重反復深化の中で利用する。つまり、各ノードの探索では、まず浅めの探索を実行して得られた指し手が最も有望だとして子ノードを有望な順に並び替える。このとき、浅めの探索の評価値を用いて最も有望な子ノードの評価値を正規分布で予測する。これにより他の子ノードの探索窓の α と β を正規分布で推定する。

探索窓の推定の詳細を述べる。提案手法では、各ノード n は 8 個の値からなる窓情報を持つ。それぞれ、並列探索窓 (α_n, β_n)、逐次探索窓 (α'_n, β'_n)、推定探索窓の正規分布の平均 $(\mu^\alpha_n, \mu^\beta_n)$ と標準偏差 $(\sigma^\alpha_n, \sigma^\beta_n)$ である。並列探索窓は、並列探索において、既に得られた結果から枝刈りの有無を判定するために用いる探索窓である。これは、子ノードが終了して得られた評価値を兄ノードも含めた残りの子ノード全ての探索窓に反映 (**図 2**) して得られる探索窓である。今回、この並列探索窓は提案手法の確率の計算には用いないこととした。逐次探索窓は、ゲーム木を逐次探索と同じ順序でノード n まで辿ったときに通過するノードのうち、探索が終了したノードの結果を用いて求めた探索窓である。つまり、各子ノードの探索が終了したときに、得られた評価値を弟ノードのみの探索窓に反映 (**図 3**) して得られる探索窓である。並列探索窓と逐次探索窓の関係として、$\alpha'_n \le \alpha_n$ かつ $\beta_n \le \beta'_n$ が満たされる。逐次探索窓を提案手法の確率の計算に用いる。

ゲーム木中のノードの窓情報の求め方を述べる。基本的にはネガマックス形式の $\alpha\beta$ 探索の探索窓の計算と同様である。今ノード n の窓情報が分かっているとする。n の浅めの探索では元々の窓情報と同じ窓情報を用いる。浅めの探索の終了後は、n の子ノードを有望な順に $c_1, c_2, \ldots, c_k$ とすると、全ての子ノード c_i について、$\mu^\beta_{c_i}$ と $\sigma^\beta_{c_i}$ 以外の窓情報は式 (1) から式 (6) となる。v_i は c_i の探索が既に終了している場合はその評価値、そうでない場合は $-\infty$ とする。図 2 と図 3 に、式 (2) と式 (4) による窓情報の更新をそれぞれ示す。

$$\alpha_{c_i} = -\beta_n \tag{1}$$

$$\beta_{c_i} = -\max\{\alpha_n, \max_{j\neq i}\{v_j\}\} \tag{2}$$

$$\alpha'_{c_i} = -\beta'_n \tag{3}$$

$$\beta'_{c_i} = -\max\{\alpha'_n, \max_{j<i}\{v_j\}\} \tag{4}$$

$$\mu^\alpha_{c_i} = -\mu^\beta_n \tag{5}$$

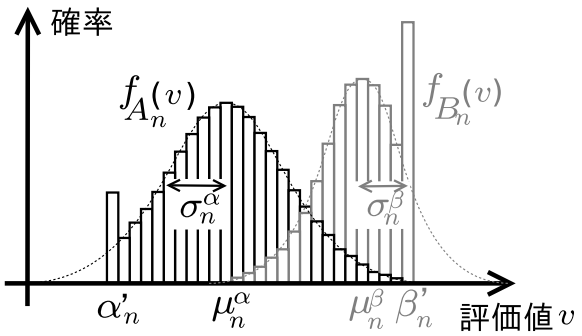

図 5 探索窓の確率分布の例

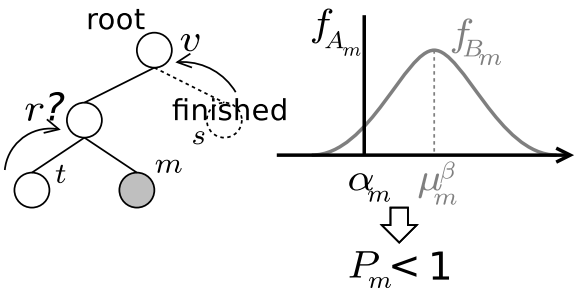

図 6 枝刈りされない確率の計算に並列探索窓ではなく逐次探索窓を用いる理由

$$\sigma_{c_i}^\alpha = \sigma_n^\beta \tag{6}$$

$\mu_{c_i}^\beta$ と $\sigma_{c_i}^\beta$ については、最左の子ノード c_1 では、$\mu_{c_1}^\beta = -\mu_n^\alpha$ かつ $\sigma_{c_1}^\beta = \sigma_n^\alpha$ とする。次に c_1 の評価値は、n の浅めの探索の評価値 r_n(c_1 の浅めの探索の評価値に等しい) から ε_n だけずれた周りに標準偏差 σ_n^r で正規分布すると予測する。今回は簡単に c_1 の探索終了までは $r_n + \varepsilon_n$ を c_1 の予測評価値とみなし (**図 4**)、最左以外の子ノード c_i では、$\mu_n^\alpha < r_n + \varepsilon_n$ ならば、最左の評価値の予測分布の方を採用し、$\mu_{c_i}^\beta = -(r_n + \varepsilon_n)$ かつ $\sigma_{c_i}^\beta = \sigma_n^r$ とする。逆に $\mu_n^\alpha > r_n + \varepsilon_n$ ならば、もともとの β の分布の方を採用し、$\mu_{c_i}^\beta = -\mu_n^\alpha$ かつ $\sigma_{c_i}^\beta = \sigma_n^\alpha$ とする。c_1 の探索終了後は c_1 の予測はもう必要ないので、$\mu_{c_i}^\beta = -\mu_n^\alpha$ かつ $\sigma_{c_i}^\beta = \sigma_n^\alpha$ とする。以上から、ルートノードで $\alpha_n = \alpha'_n = \mu_n^\alpha = -\infty$、$\beta_n = \beta'_n = \mu_n^\beta = \infty$ とすれば、全てのノードの窓情報が再帰的に求まる。

今回は簡単に、正規分布に従う二つの確率変数の最大値の分布を平均が大きい方の分布としたが、それぞれの平均と標準偏差から最大値の分布の平均と標準偏差を計算する方法 [2] も試した。しかし、実験結果に有意な差は見られなかった。

2.2 枝刈りされない確率の計算

ノード n の探索窓 α と β を確率分布で予測できれば、ノード n が枝刈りされない確率 P_n を計算することができる。これは n で $\alpha < \beta$ となる確率なので、α と β の予測分布が独立だと仮定すると次式となる。

$$\begin{aligned} P_n &= P(A_n < B_n) \\ &= \sum_{b=-\mathrm{INF}}^{\mathrm{INF}} \left(f_{B_n}(b) \sum_{a=-\mathrm{INF}}^{b-1} f_{A_n}(a) \right) \end{aligned} \tag{7}$$

ただし、A_n と B_n は n の α と β を表す確率変数、INF は十分大きい整数とする。f_{A_n} と f_{B_n} は A_n と B_n の確率質量関数[*3]である。評価値や探索窓の値は整数なので密度関数ではなく質量関数で考える。α や β は既に分かっている逐次探索窓 (α'_n, β'_n) の外側の値になることはないと考えられるため、$f_{A_n}(a)$ と $f_{B_n}(b)$ は以下のように正規分布が切れた分布で表現できる。**図 5** に分布の例を示す。

$$f_{A_n}(a) = \begin{cases} 0 & (a < \alpha'_n) \\ \sum_{a=-\mathrm{INF}}^{\alpha'_n} \frac{1}{\sqrt{2\pi}\sigma_n^\alpha} \exp\left(\frac{-(a-\mu_n^\alpha)^2}{2(\sigma_n^\alpha)^2}\right) & (a = \alpha'_n) \\ \frac{1}{\sqrt{2\pi}\sigma_n^\alpha} \exp\left(\frac{-(a-\mu_n^\alpha)^2}{2(\sigma_n^\alpha)^2}\right) & (a > \alpha'_n) \end{cases} \tag{8}$$

$$f_{B_n}(b) = \begin{cases} 0 & (b > \beta'_n) \\ \sum_{b=\beta'}^{\mathrm{INF}} \frac{1}{\sqrt{2\pi}\sigma_n^\beta} \exp\left(\frac{-\left(b-\mu_n^\beta\right)^2}{2\left(\sigma_n^\beta\right)^2}\right) & (b = \beta'_n) \\ \frac{1}{\sqrt{2\pi}\sigma_n^\beta} \exp\left(\frac{-\left(b-\mu_n^\beta\right)^2}{2\left(\sigma_n^\beta\right)^2}\right) & (b < \beta'_n) \end{cases} \tag{9}$$

逐次探索で推定される最小探索木に含まれるノード m では、$\mu_m^\alpha = \alpha'_m = -\infty$ または $\mu_m^\beta = \beta'_m = \infty$ が成り立つので、$P_m = 1$ となる。

ここで、枝刈りされない確率の計算に並列探索窓 (α_n, β_n) ではなく、逐次探索窓 (α'_n, β'_n) を用いた理由を述べる。**図 6** において、ノード m は予測最小探索木に含まれる。しかし、並列探索窓を用いて提案手法の枝刈りされない確率を計算すると、図中のノード s が終了し、かつノード t がまだ終了していない場合、$\mu_m^\beta = -r$、$\alpha_m = v$ から $P_m < 1$ となり、ノード m が枝刈りされることがあると判断する。ここでは今回は最小探索木の予測を優先し、ノード m が枝刈りされると予測されることがないように、並列探索窓の代わりに逐次探索窓を用いた。これはヒューリスティックではあるが、実際に評価では、並列探索窓を用いるより逐次探索窓を用いた方がよい性能が得られた。

2.3 表引きによる高速化

式 (7) の計算量は評価値が取りうる値の数を N とすると、$O(N^2)$ となり、計算コストが大きい。そこで、実装では各パラメータに応じて予め式 (7)(8)(9) を用いて確率表を作り、探索時には表引きで確率を求めた。用いるパラメータの数を減らすため、$\mu_n^\alpha \to 0$、$\sigma_n^\alpha \to 100$ に正規化し、以下の 4 つのパラメータを用いて表引きを行った。

$$\eta_n^\beta = (100/\sigma_n^\alpha)(\mu_n^\beta - \mu^\alpha) \tag{10}$$

[*3] 例えば、$f_{A_n}(a) = P(A_n = a)$ である。これはノード n の α の値が a である確率である。

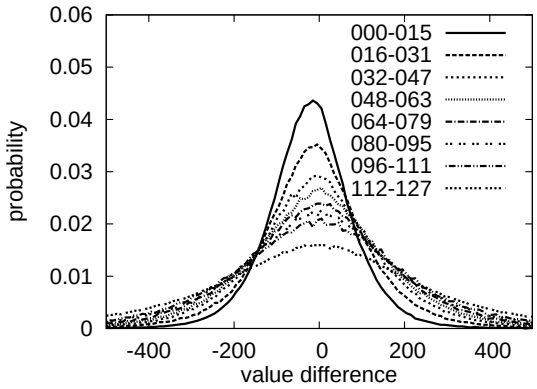

図 7 (深さ 10 の評価値)-(深さ 8 の評価値) の進行度ごとの分布

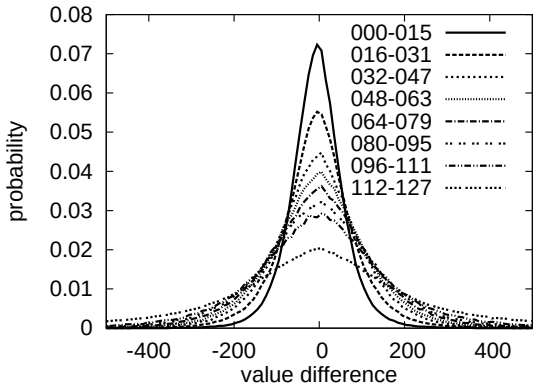

図 8 (深さ 16 の評価値)-(深さ 14 の評価値) の進行度ごとの分布

$$\rho_n^\beta = (100/\sigma_n^\alpha)\sigma_n^\beta \tag{11}$$

$$\delta_n^\alpha = (100/\sigma_n^\alpha)(\alpha'_n - \mu_n^\beta) \tag{12}$$

$$\delta_n^\beta = (100/\sigma_n^\alpha)(\beta'_n - \mu_n^\alpha) \tag{13}$$

式 (12) や式 (13) を用いたのは、α'_n は μ_n^β に近いとき、β'_n は μ_n^α に近いときが、パラメータが変化したときに確率が大きく変動するため重要だからである。なお、各パラメータの値は少しずつ集約して表が大きくなりすぎないようにした。具体的には η_n^β と δ_n^α と δ_n^β が -250 から 250 までの 5 刻みごと、δ_n^β が 50 から 200 までの 5 刻みごとに集約した。

3. 異なる深さでの評価値の差の分布の調査

提案手法では各局面について、浅い探索の評価値から実際に行う探索の評価値を正規分布で予測する。具体的には節 2.1 で述べた浅い探索の評価値と予測分布の平均とのずれ ε_n と分布の標準偏差 σ_n^r を推定しなければならない。これを実現するために、プロの棋譜に現れる各局面を将棋プログラムである激指[*4]を用いて深さ 8、10、12、14、16 で探索し、ある深さの評価値からそれより 2 浅い探索の評価値を引いた差の分布を調べた。つまり、16-14、14-12、12-10、10-8 の 4 通りである。後手番の評価値は符号を反転して、それぞれの評価値は自分の手番から見て有利なほど大きいものとした。用いた棋譜数は 3 万である。初期局面など同じ局面の重複は除いた。さらに詰みが見つかったときの評価値の絶対値は非常に大きく、平均や標準偏差の計算に大きな影響を与えるため、詰みが見つかった局面も除いた。

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

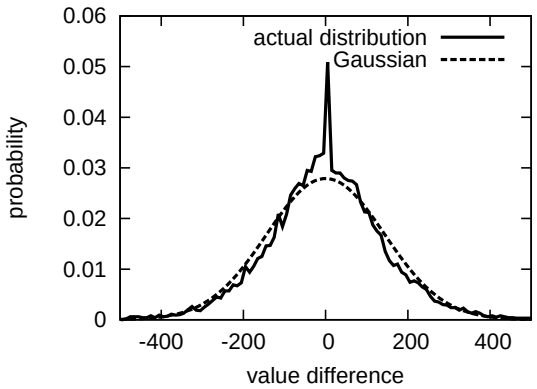

図 9 進行度が 48 から 63、深さ 10 の評価値の絶対値が 200 から 299 の局面について、深さ 12 と深さ 10 の評価値の差の分布

局面により、ε_n や σ_n^r は異なると考えられる。そこで、激指の進行度と、浅めの探索の評価値の絶対値の 2 つを局面の特徴とした。進行度と評価値は明らかに独立ではないが、簡単に用いることができる量として今回はこれらの 2 つを用いた。激指の進行度は 0 から 127 の整数であり、値が大きいほど終盤であることを示す。これを進行度 16 ごとに 8 つの区分に分けて分布を調べた。同様に、評価値の絶対値も 7 つの区分に分けて分布を調べた。評価値の絶対値は進行度が大きいほど大きくなりやすいため、進行度によって、この 7 つの区分の分け方は変更した。合計で $4 \times 8 \times 7 = 224$ 組の平均と標準偏差が得られた。

得られた分布に関して、以下の傾向があった。

- 評価値の差は 0 が例外的に出現頻度が大きい。周りの点の 3 倍から 10 倍程度の多さになっていた。
- 進行度が小さいほど標準偏差が小さい。
- 評価値の絶対値が小さいほど標準偏差が小さい。
- 深さが大きいほど標準偏差が小さい。ただし例外的に、進行度も評価値の絶対値も大きいときは、深さが大きいほど標準偏差が大きい。
- 進行度が小さく、深さが小さいほど、浅い探索が楽観的評価 ($\varepsilon_n = -10$ 程度) をする傾向にあるが、多くの場合は ε_n はほぼ 0 である。

深さ 10 と深さ 8 の評価値の差の分布を**図 7** に、深さ 16 と深さ 14 の評価値の差の分布を**図 8** に示す。横軸は評価値の差、縦軸は各出現回数を総数で割って求めた確率である。横軸は 10 ごとに和をとってまとめてプロットしている (例えば横軸が 5 の点は評価値差が 1 から 10 までとなる確率の和)。また差が 0 になる点は図の見やすさのために除外している。これらの図から、上で述べた分布の傾向の一部を確認できる。

また、求めた ε_n と σ_n^r の正規分布で実際の分布がどのくらい予測できているのかを確認する。各パラメータがおよそ中間の値をとる例として、進行度が 48 から 63 の間で、かつ深さ 10 の評価値の絶対値が 200 から 299 の間の局面を、深さ 12 と深さ 10 で探索したときの評価値の差の分布と、そのときの予測正規分布を同時に**図 9** に示す。この場合は差が 0 付近の部分を除いて正規分布で近似できていることが分かる。

4. 評価

提案手法を評価するために、次の 3 つの探索順序を比較した。なお、全ての手法において、P_n が同じノードはゲーム木の左側にあるノードから順に探索 [9] した。

- **提案手法**
  P_n が大きいものから順に実行
- **分布なし**
  提案手法において、
  $$\max\{\mu_n^\alpha, \alpha_n'\} < \min\{\mu_n^\beta, \beta_n'\} \tag{14}$$
  となるノード n は $P_n = 1$、それ以外のノードは $P_n = 0$ として実行
- **深さ優先**
  提案手法の計算で $P_n = 1$ となるノードは優先して実行[*5]し、それ以外のノードはゲーム木の左側にあるノード、つまり深さ優先探索で先に探索されるノードから順に実行

深さ優先の手法でも提案手法による確率の計算は行っている。これは実装を統一して実験を簡単に行うためである。したがって、不必要な計算を行っていることになるが、探索順序における性能差を比較するためにはこれでよいと考えた。また、比較として分布を用いない場合も評価する。これは、確率分布を用いずに、単純に予測評価値を用いて探索を進めた場合に、枝刈りされると予測されるノードは単に後回しにすればよいという手法である。ただし、浅い探索の評価値を用いて深い探索の評価値を予測するため、今回は最左の子ノードの評価値の予測しか用いていない。

4.1 将棋のゲーム木の実探索による評価

並列ゲーム木探索のプログラムに提案手法を組み込んで、探索時間を測定した。並列探索の実装は [12] と同じものを用いた。マスタ・ワーカ方式で、マスタが $\alpha\beta$ 探索を行い、残り深さがある閾値 g 以下になるとタスクとして部分木をワーカに送信し、ワーカが探索して結果を返す。なお、確率が $P_n = 1$ であるノードは、$Pn < 1$ のノードの探索を中断してでも探索される [11]。ただし、実装の簡略化のため、$P_n = 1$ だったノードが $P_n < 1$ に変化したとしても、タスクとしてワーカに既に送ってしまった場合は、ワーカに $P_n < 1$ になったことは伝えないものとした。

探索中における確率の更新では、確率が変化しないと分かっている部分木のノードの更新を行う必要がない。しかし、そのような最適化はまだ行っておらず、部分木をすべて辿っているため、実装にはまだ無駄が残っていることを述べておく。探索窓の分布予測には節 3 の結果の ε_n と σ_n^r 用いた。ただし、深さ 14 以上の部分木については深さ 14 だとして確率を計算している。

実験に用いた局面は、プロ棋士の 100 棋譜から 1 棋譜につき 1 局面ずつ重複しないようにランダムに計 100 局面選び、激指の進行度を用いて序盤と終盤の局面は除いた。各局面 5 回ずつ探索し、500 回の探索での実行時間の平均と標準偏差を求めた。実験環境は以下の通りである。各ノードにつき 8 ワーカを実行し、全部で 512 ワーカで実験した。

- 1 ノードあたり 4 コア ×2 スレッド ×2 ソケット
- CPU : Xeon E5530 2.40GHz
  - 8M cache, 2.40GHz, 5.86 GT/s QPI
- 1 ノードあたりのメモリ : 24GB
- ネットワーク : 10Gbps Ethernet
- OS : Linux 2.6.32-5
- コンパイラ : gcc 4.4.5

探索時間が短くなるようなパラメータ設定として、探索深さが 18 で閾値 g が 12 のときと、探索深さが 22 で g が 13 のときで実験を行った。ただし、探索深さが 22 のときは実験時間の関係で 30 局面だけを用いた。結果を**表 1** と**表 2** にそれぞれ示す。探索順序によって明確な差がついていないことが分かる。

4.2 将棋のゲーム木を用いたシミュレーション

提案手法をより理想的な場合について評価するため、[12] と同じシミュレーション方法を用いた。これはリーフの評価にかかる時間、つまり激指の探索関数の計算時間が 1 ステップで全て等しく、それ以外の処理にかかる時間を全て無視したものである。p ワーカでの並列探索のシミュレーションでは、p 個のリーフを選択するまで木を展開し、そ

表 1 将棋を用いた深さ 18 の実探索の実行時間 [秒]

探索順序	平均実行時間	標準偏差
提案手法	7.66	4.63
深さ優先	7.61	4.77
分布なし	7.43	4.70

表 2 将棋を用いた深さ 22 の実探索の実行時間 [秒]

探索順序	平均実行時間	標準偏差
提案手法	141	141
深さ優先	136	123
分布なし	141	138

[*5] 実際の実装では確率が 1 に十分近ければ 1 とみなす

れらのリーフを評価するという操作を 1 ステップとする。このステップを繰り返し、探索終了までのステップ数を実行時間とみなす。

リーフは実際の並列探索では部分木であり、この探索時間は実際には大きくばらつく上、探索窓によっても大きく異なる。これを全て同じ時間しかかからないとみなすのは非常に乱暴である。したがって、このシミュレーションは実際の並列探索をシミュレーションしているわけではない。つまり、このシミュレーション方法で性能が良かったからといって実際の並列探索で性能が良いとは限らない。しかし、各ワーカが評価関数を呼んだ回数を数えていくというシミュレーションは非常に時間がかかると思われるため、今回のシミュレーション方法をとった。理想的な場合でも性能が良くなければ、実探索で性能が良いことは期待できないため、理想的な場合の性能を確認する意味はあると考えた。

実験に用いた局面は節 4.1 で用いた 100 局面を用いた。この 100 局面での実行ステップ数の平均と標準偏差を求めた。本論文では大規模環境を想定しているため、結果はプロセス数が多い場合を中心に**表 3** に示す。深さ優先を比較すると提案手法の方が実行ステップ数が少なく、台数が多いと 10%から 15%の差がついたが、提案手法と分布なしを比較すると性能差は小さかった。

4.3 人工木を用いたシミュレーション

将棋のゲーム木での結果を考察するため、提案手法を人工木を用いてさらに理想的な状況で評価した。人工木は全てのノードでミニマックス値が浅めの探索なしで正規分布で予測でき、その標準偏差は全てのノードで等しいと近似的にみなせる木を作成して用いた。各ノードでの子ノードの並び替えは、予測正規分布の平均が大きいものをより有望な子ノードとして扱うことで行った。

用いた人工木の作り方を述べる。木のエッジに乱数を与えて、ルートからそのノードまでのエッジの値の和がそのノードの評価値となるような人工木を作る。この人工木は先行研究でも評価に用いられている [6], [8]。この人工木の各ノードの評価値の分布を予測するというのが基本的な考え方である。まず、親ノードと子ノード b 個からなる小さな部分木を考える。親ノードの暫定評価値を 0 とする。i 番目の子ノードの評価値は、独立同分布な確率分布 P_S から得られる乱数を、親ノードの暫定評価値に加えたものであるという木を考える。この乱数の確率関数 $f_S(s)$ は既知とし、i 番目の子ノードに加えられた値を表す確率変数を S_i とする。子ノード間の評価値は実際には独立ではないと考えられる。しかし、このモデルは評価値の確率分布が与えられている理想的な場合における本提案手法の評価には有効だと考えた。今、V_i は i 番目の子ノードの評価値を表す確率変数とする。$S_i = s_i$ で条件付けられた各子ノードの確率関数 $f_{V_i|S_i}(v_i|s_i)$ は $f_{V_i|S_i}(v_i|0)$ を s_i だけシフトしたものだとする。

$$f_{V_i|S_i}(v_i|s_i) = f_{V_i|S_i}(v_i - s_i|0) \tag{15}$$

さらに $f_{V_i|S_i}(v_i|0)$ は全ての子ノードで共通で $f_{V|S}(v|0)$ とし、これは既知だとする。

このとき、各子ノードの s_i が未知のときの親ノードの評価値、すなわち、子ノードの評価値の最大値 X の確率関数 $f_X(x)$ が知りたい。これは以下のように、$f_X(x, s_1, \ldots, s_b)$ を全ての s_i で周辺化することによって得られる。

$$\begin{aligned} & f_X(x) \\ &= \sum_{s_1=-\mathrm{INF}}^{\mathrm{INF}} \cdots \sum_{s_b=-\mathrm{INF}}^{\mathrm{INF}} f_{X,S_1,\ldots,S_b}(x, s_1, \ldots, s_b) \\ &= \sum_{s_1=-\mathrm{INF}}^{\mathrm{INF}} \cdots \sum_{s_b=-\mathrm{INF}}^{\mathrm{INF}} \\ &\qquad \left(f_{X|S_1,\ldots,S_b}(x|s_1, \ldots, s_b) \prod_{i=1}^{b} f_{S_i}(s_i) \right) \end{aligned} \tag{16}$$

二つ目の式変形では s_i が他の子ノードとは独立に決まることを利用している。$f_{X|S_1,\ldots,S_b}(x|s_1, \ldots, s_b)$ は、子ノードの s_i が既知のときに、評価値の最大値が x になる確率である。これは、全ての子ノードの評価値が x 以下である確率から $x-1$ 以下である確率を引いたものなので、

$$\begin{aligned} & f_{X|S_1,\ldots,S_b}(x|s_1, \ldots, s_b) = \\ & \prod_{i=1}^{b} \sum_{y=-\mathrm{INF}}^{x} f_{V_i|S_i}(y|s_i) - \prod_{i=1}^{b} \sum_{y=-\mathrm{INF}}^{x-1} f_{V_i|S_i}(y|s_i) \end{aligned} \tag{17}$$

で表せる。このとき、

$$f_{V_i}(v_i) = \sum_{s_i=-\mathrm{INF}}^{\mathrm{INF}} f_{V_i|S_i}(y - s_i|0) f_{S_i}(s_i) \tag{18}$$

であるので、式 (16)(17)(18) から以下となる。

$$f_X(x) = \prod_{i=1}^{b} \sum_{y=-\mathrm{INF}}^{x} f_{V_i}(y) - \prod_{i=1}^{b} \sum_{y=-\mathrm{INF}}^{x-1} f_{V_i}(y) \tag{19}$$

つまり、式 (18) で $f_S(s)$ と $f_{V|S}(v|0)$ から $f_V(v)$ を求めた後で、式 (19) から $f_X(x)$ を求めればよい。

次にこの計算を繰り返す。$f_S(s)$ は平均 0、標準偏差 100 の

表 3 将棋を用いたシミュレーションの実行ステップ数

プロセス数	1	256	1024	4096	16384
提案手法	114536	619	191	75.3	42.7
(標準偏差)	57770	334	78.8	24.5	11.9
分布なし	114536	649	196	80.0	47.3
(標準偏差)	57770	325	88.5	28.3	13.6
深さ優先	114536	642	196	84.6	51.2
(標準偏差)	57770	322	90.3	27.0	16.0

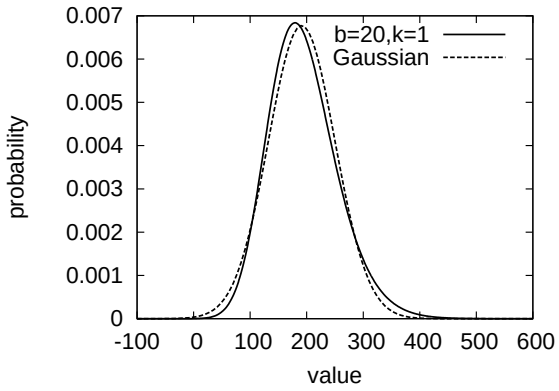

図 10 分枝数 20、残り深さ 1 のときの人工木の評価値の分布

正規分布とする。$f_{V|S}(v|0)$ の初期値 $f^{(0)}_{V|S}(v|0)$ は $v=0$ で 1、それ以外で 0 となる分布とする。このとき、$f^{(k)}_{V|S}(v|0)$ を既知として、親ノードの $f_X(x)$ を求め、$f^{(k+1)}_{V|S}(v|0) = -f_X(v)$ とする。以降はこの計算を繰り返す。$f^{(k)}_{V|S}(v|0)$ はあるノードから深さ k だけ探索したときの評価値の分布を表している。このとき各 k について、$f^{(k)}_{V|S}(v|0)$ の平均と標準偏差を調べていくと、k が増えるにつれてある値に収束しているように見えた (平均については k の偶奇で異なる) ため、$f^{(50)}_{V|S}(v|0)$ の平均と標準偏差を ε_n と σ^r_n として用いることとした。$b=5$ のときの ε_n は k が偶数と奇数のときにそれぞれ 26.2 と 125.5、σ^r_n は k の偶奇に依らず 85.6、$b=20$ のときの ε_n が 22.6 と 191.1、σ^r_n が 58.9 であった。実際に人工木を構築するときは、エッジに乱数を与える方法で人工木を作り、各ノードの評価値を予測評価値とする。さらに、リーフノードの評価値に $f^{(50)}_{V|S}(v|0)$ から得られる乱数を与え、これを実際の評価値とする。$b=20$ のときの残り深さが 1 のノードのミニマックス値の分布と予測正規分布と合わせて**図 10** に示す。実際の分布は正規分布ではないが、正規分布で近似できていることが分かる。

人工木を用いた評価では多重反復深化の計算コストも考慮するため浅めの探索も行った。ただし、この浅めの探索は各ノードのミニマックス値や子ノードの並び替えは全て同じで探索する深さだけが違う擬似的なものであり、浅めの探索の結果は捨てるものとした。さらに、トランスポジションテーブルを用いて、既に探索したノードに関しては重複して探索しないようにしている。トランスポジションテーブルに用いるノードのキーとしては、並列探索を深さ優先探索で行うためにも利用している signature [5], [9] を用いた。

提案手法では、(1) 最左の子ノードの評価値の予測しか用いていない点 (2) 正規分布に従う二つの確率変数の最大値の分布は正規分布にならないが、それを平均が大きい方の正規分布で近似している点 (3) 粒度の粗い表引き、の 3 つの高速化により確率推定の精度が落ちている可能性がある。そこで、これらの高速化を行わずに確率を精度良く求める手法として、α と β の予測分布を平均と標準偏差で管理するのではなく、各値ごとの確率 (ヒストグラム) で管理し、さらに最左の子ノードだけではなく全ての子ノードの評価値の予測分布を用いる手法 (高速化なし) を準備した。

分枝数と探索深さがそれぞれ 5 と 10 の結果を**表 4** に、20 と 6 の結果を**表 5** に示す。100 個の人工木での実行ステップ数の平均と標準偏差を求めた。表 4 で提案手法を深さ優先と比べると、1024 台で性能差が最大となりステップ数が 54%になっている。台数が少ないと確率 1 のタスクだけで十分な数があるため性能に差がつかない。台数が多いと性能差が小さくなるのは、浅めの探索の終了を待つために各時点で実行可能なタスクの数が限られ、各ステップでそれらが全て実行できることが多くなるためである。実際に浅めの探索なしで実験したところ、台数が増えるほど提案手法と比較手法の性能差は大きくなった。提案手法と分布なしを比較しても最大でステップ数が 82%と提案手法の有効性が見られる。次に表 5 を見ると、表 4 と比べて各手法間で性能差が小さくなっている。これは分枝数が多いと確率 1 のタスクが多くなるためだと考えられる。最後に、提案手法でヒストグラムを保持して、より正確に確率を計算しても、実行ステップ数の減少は 5%から 10%程度であるが、実際の計算時間を調べたところほとんどの場合で 10 倍以上であり、100 倍以上であることもあった。これに対し、提案手法と分布なしの実際の計算時間は同程度であった。

表 4 人工木でのシミュレーションの実行ステップ数 (分枝数 5、探索深さ 10)

プロセス数	1	256	1024	4096	16384
高速化なし	23478	145	62.4	39.5	30.6
(標準偏差)	5551	31	8.9	2.5	1.2
提案手法	23478	158	65.8	41.6	32.5
(標準偏差)	5551	38	9.9	3.4	1.7
分布なし	23478	153	75.2	50.9	36.3
(標準偏差)	5551	36	13.3	7.6	3.9
深さ優先	23478	237	121	65.7	39.8
(標準偏差)	5551	70	35	15.2	6.1

表 5 人工木でのシミュレーションの実行ステップ数 (分枝数 20、探索深さ 6)

プロセス数	1	256	1024	4096	16384
高速化なし	35037	154	48.4	20.4	14.2
(標準偏差)	7508	32	8.8	2.6	0.5
提案手法	35037	174	54.0	22.1	15.0
(標準偏差)	7508	41	12.7	3.5	1.4
分布なし	35037	166	53.1	23.0	16.4
(標準偏差)	7508	39	11.5	4.1	2.1
深さ優先	35037	174	59.7	31.5	22.7
(標準偏差)	7508	42	15.5	8.0	4.8

4.4 考察

提案手法が実探索で有効でなかった理由を考える。まず、実探索の実験でのワーカ数が少なかったと考えられる。シミュレーションでも 1024 台で 3%しか差がなかったので、それより少ない 512 台の実探索でその差が見えなくても不思議ではない。次に、表 3、表 4、表 5 を見比べると、将棋でのシミュレーションの結果は分枝数が 5 のときよりも 20 のときの結果に近いと言え、さらに人工木を用いた場合の各探索順序の性能差も分枝数が大きくなると小さくなったことから、分枝数が大きいという将棋の特徴が影響している可能性が考えられる。

また、分布を用いる提案手法と分布を用いない手法では、深さ優先と大きな差があった分枝数が 5 の人工木でも差が小さかった。提案手法で確率が高いノードは、分布を用いずに予測値を用いても枝刈りされないと判断されるため、この二つでは探索するノードがあまり変わらないからだと考えられる。

5. 関連研究

ゲーム木探索において確率分布を用いた研究として、モンテカルロ木探索において勝率を分布で予測したものがある [10]。分布には正規分布を用いており、正規分布に従う子ノードの勝率の最大値の分布を正規分布で予測するために、二つの正規分布の平均と標準偏差から、最大値の分布の平均と標準偏差を計算する方法 [2] を用いている。

並列探索において、評価値を予測して探索を進める手法として APHID [1] の guessed score と探索窓選択がある。APHID はマスタ・ワーカ方式で、リーフノードの探索をワーカに割り当てる。ワーカは割り当てられたノードの探索を反復深化で行っていく。マスタはワーカが探索しているリーフノードの評価値を用いながら探索を進めるが、ワーカの探索が必要な深さまでまだ終了していない場合は、既に終了している深さの探索の評価値から必要な深さの評価値を予測 (guessed score) して探索を進め、予測値を使わなくなるまでマスタの探索を繰り返す。予測値を用いた探索で枝刈りされると判断されているノードは、ワーカに割り当てられていたとしても、ワーカは探索を実行しない。これは、実際にはまだ枝刈りされていないノードでも、枝刈りされると予測されるノードの優先度を最下位にして探索を進める手法であると言える。また、ワーカはマスタから guessed value を受け取り、これによりノードを探索する際の探索窓を決定する。

6. おわりに

本論文では、枝刈りされない確率が高いノードから探索をしていくという並列探索手法のために、各ノードの探索窓の α と β を確率分布で予測し、それらを用いて枝刈りされない確率を計算するという方法を提案した。分枝数が少ない人工木の場合は提案手法の有効性が見られたが、分枝数が多い場合や、将棋の場合は有効性を確認することができなかった。

今後の課題としては、ゲーム木の分枝数が大きな影響を与えていることが考えられたので、分枝数が小さいゲームで評価する、あるいは、将棋でも人工的に分枝数を制限して評価するといったことを行いたい。さらに、多重反復深化による同期も提案手法の効果に影響を与えることが分かったので、浅い探索の結果を用いることなく、静的評価値などから分布を予測することにより、多重反復深化を用いないプログラムに対しても評価を行いたいと考えている。

謝辞 本研究の一部は、総務省委託研究「広域災害対応型クラウド基盤構築に向けた研究開発 (高信頼クラウドサービス制御基盤技術)」の一環として実施された。

参考文献

[1] Brockington, M. and Schaeffer, J.: APHID: Asynchronous parallel game-tree search, *Journal of Parallel and Distributed Computing*, Vol. 60, No. 2, pp. 247–273 (2000).

[2] Clark, C.: The greatest of a finite set of random variables, *Operations Research*, Vol. 9, No. 2, pp. 145–162 (1961).

[3] Feldmann, R.: Game Tree Search on Massively Parallel Systems, PhD Thesis, University of Paderborn (1993).

[4] Himstedt, K., Lorenz, U. and Moller, D.: A Twofold Distributed Game-Tree Search Approach Using Interconnected Clusters, *Euro-Par '08*, Vol. 5168, Springer, pp. 587–598 (2008).

[5] Kishimoto, A. and Schaeffer, J.: Distributed game-tree search using transposition table driven work scheduling, *ICPP '02*, IEEE, pp. 323–330 (2002).

[6] Korf, R. and Chickering, D.: Best-first minimax search, *Artificial Intelligence*, Vol. 84, No. 1-2, pp. 299–337 (1996).

[7] Kuszmaul, B.: The StarTech massively parallel chess program, *International Computer Chess Association Journal*, Vol. 18, No. 1, pp. 3–19 (1995).

[8] Shoham, Y. and Toledo, S.: Parallel Randomized Best-First Minimax Search, *Artificial Intelligence*, Vol. 137, No. 1-2, pp. 165–196 (2002).

[9] Steenhuisen, J.: Transposition-Driven Scheduling in Parallel Two-Player State-Space Search, Master's thesis, Delft University of Technology (2005).

[10] Tesauro, G., Rajan, V. and Segal, R.: Bayesian Inference in Monte-Carlo Tree Search, *UAI 2010* (2010).

[11] 浦晃，横山大作，近山隆：投機を用いた並列ゲーム木探索の効率化，第 15 回ゲームプログラミングワークショップ，pp. 134–141 (2010).

[12] 浦晃，三輪誠，横山大作，田浦健次朗，近山隆：探索が必要となる確率を用いた並列 α β 探索のスケジューリング，第 16 回ゲームプログラミングワークショップ，pp. 68–75 (2011).