局面の実現確率に基づく ゲーム木探索アルゴリズム

鶴岡慶雅[†]　横山大作[†]　丸山孝志[†]　近山隆[‡]

[†] 東京大学工学系研究科
[‡] 東京大学新領域創成科学研究科

現在のトップレベルのコンピュータ将棋プログラムの多くは深さ打ち切りのミニマックス法を基本としている．深さ打ち切りによる探索は，以前から多くの欠点を指摘されながらも，依然としてコンピュータ将棋における最も実用的な探索手法とされている．それに対して本論文では，局面の実現確率を打ち切り条件とする探索アルゴリズムを提案する．局面の実現確率とは，その局面にいたるまでの手順が実際に指されて，その局面が実現する確率を表す．したがって，探索打ち切り条件を局面の実現確率とすることで，「ありそうな展開」を中心に読むことができ，効率的な探索が可能になることが期待される．本論文では，提案アルゴリズムを実装した将棋プログラムを作成し，深さ打ち切りアルゴリズムとの比較による評価を行なった．実験の結果，実現確率打ち切りプログラムは，深さ打ち切りプログラムよりも有意に強く，その効果は，深さ打ち切りプログラムを5倍に高速化する効果に相当することを確認した．

Game-Tree Search Algorithm Based on Realization Probability

Yoshimasa Tsuruoka[†]　Daisaku Yokoyama[†]　Takashi Maruyama[†]
Takashi Chikayama[‡]

[†] Faculty of Engineering, The University of Tokyo
[‡] School of Frontier Sciences, The University of Tokyo

Most of the state-of-the art computer shogi programs adopt variants of the fixed-depth minimax search algorithm. Although many drawbacks of the fix-depth search have been pointed out, the algorithm is still considered to be the most practical one for computer shogi programs. In this paper we propose a new game tree search algorithm based on realization probability instead of depth. Realization probability of a certain node represents the probability that the moves leading to the node will actually to be played. Thus, this algorithm spends little computational resource on unrealistic moves, resulting in more effective search. We provide experimental results showing that this new algorithm performs significantly better than the fixed-depth search algorithm and its performance gain is equivalent to speedup of more than 5 times.

1 はじめに

コンピュータチェスでは，Deep Blue が深さ打ち切りの全幅探索という，いわば力まかせの探索手法により人間のチャンピオンを凌駕するに致った．コンピュータチェスでそのようなしらみつぶし的な探索が成功した理由の一つは，チェスの分岐因子が少ないことにある．しかし将棋は持ち駒の使用というルールがあるために，分岐因子がチェスよりもはるかに大きく，コンピュータチェスと同様の力まかせの探索で高い棋力を達成することは難しい．

現在のトップレベルのコンピュータ将棋プログラムの多くは深さ打ち切りのミニマックス法を基本としている [9, 6, 5]．深さ打ち切りのミニマックス法に関しては，PVS 法や MTD(f) 法 [4] などの効率的探索手法が提案されている．

深さ打ち切りではない探索手法としては，Probabilistic B* [1] や，Alpha-beta-conspiracy search[3] などの有望な手法がいくつか提案されているが，実用的な性能という点で深さ打ち切りの α β 法と比較してはっきり優れているといえるアルゴリズムは未だ提案されていない [2]．

本論文では，局面の実現確率を打ち切り条件とする探索アルゴリズムを提案する．局面の実現確率とは，その局面にいたるまでの手順が実際に指されて，その局面が実現する確率を表す．したがって，探索打ち切り条件を局面の実現確率とすることで，実際に実現しそうな展開，すなわち「ありそうな展開」を中心に読むことができ，効率的な探索が可能になることが期待される．

本論文の構成を以下に示す．2 章で実現確率打ち切り探索アルゴリズムについて説明する．3 章で，提案アルゴリズムの評価を深さ打ち切りアルゴリズムとの対戦と次の一手問題によって行なう．4 章で実験結果についての考察を述べる．5 章でまとめを行なう．

2 実現確率打ち切りアルゴリズム

2.1 基本的な考え方

本論文で提案する実現確率打ち切りアルゴリズムでは，深さの代わりに局面の実現確率を探索の打ち切り条件とする．局面の実現確率とは，その局面にいたるまでの手順が実際に指されて，その局面が実現する確率である．いま，ある指し手によって，ある局面から別の局面に変化する確率，すなわち遷移確率が与えられているとすると，局面の実現確率は次のように再帰的に計算することができる．

（ある局面の実現確率）＝（直前の局面の実現確率）×（遷移確率）

もちろん，ルート局面はすでに実現しているのだから実現確率は 1 である．したがって，読みの中の局面の実現確率は，1 から始まって，読みが深くなるにつれて徐々に減少していくことになる（図 1）．そして，その実現確率が，閾値を下回った時点で，ノードの展開をやめて末端とする．このようなアルゴリズムにすることで，実現する確率がある一定値以上の局面，すなわち「ありそうな展開」のみを読むことができる．

探索の打ち切り条件以外は，通常の α β 法と同様である．反復深化も同様に行なう．すなわち，深さで反復深化を行なうかわりに，実現確率で反復深化を行なう．

実現確率を打ち切り条件とする理由は次のようなものである．

一つには，将棋のエキスパートプレイヤは，深さに基づいて指し手を読んでいるわけではない．「ありそうな展開」は深く読むし，逆に，「ありそうでない展開」についてはほとんど読まない．局面の実現確率は，「ありそうな」ということを定量化したものと考えることができる．したがって，それを探索の打ち切り条件とすることで「ありそうな展開」を中心に読むという，人間のエキスパートプレイヤの読みに近い思考法を実現できる可能性がある．

もう一つは，現在のトップレベルの将棋プログラムが，深さを基本的な打ち切り条件としながらも，実現する可能性の高そうな展開に対しては，探索深さを延長していることである．例えば，「IS 将棋」では，手

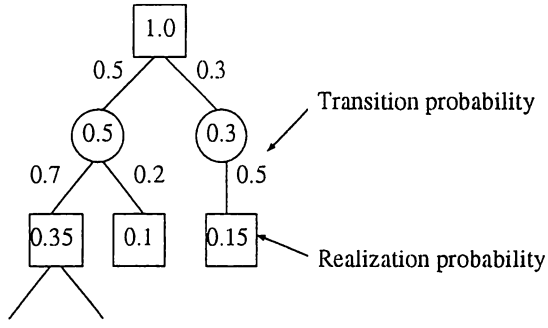

図 1: 実現確率と遷移確率

筋の一連の手順を深く読んでいるし [9],「金沢将棋」では,「王手に対する最善手」や「囲いの駒を取る手に対する最善手」などの手の探索深さを延長している [6]. また, YSS では, 浅い探索での最善手の探索深さを延長するということを行なっている [5]. なぜこのような探索延長をすることが有効なのかは必ずしも明確ではないが, これらの探索延長は,「実現する可能性の高いノードは展開した方が良い」ということを示唆しているようにも見える. それならば最初から深さを打ち切り条件とするのではなく, 実現確率自体を打ち切り条件とすれば良いのではないかということである.

実現確率打ち切りアルゴリズムは, ノードの展開順を形式的にみれば, 実現確率を評価値とした最良優先探索（正確には反復深化による最良優先探索）を行なっていると考えることもできる. しかしゲーム木探索の目的は, 当然のことながら, 実現確率が最大の局面を求めることではない. したがって, 探索の意味は全く異なったものとなっている.

2.2 カテゴリごとの遷移確率

実現確率を打ち切り条件とする探索アルゴリズムを実装する上で問題になるのが, ある局面が別の局面に変化する確率である「遷移確率」をどのようにして求めるかということである. 本アルゴリズムでは, 基本的には, 遷移確率を指し手の表面的な性質に基づいて算出する.

具体的には, 指し手の性質を数十種類のカテゴリに分け, プロ棋士の実戦棋譜（「羽生善治実戦集」に含まれている約 600 局）から統計情報を抽出することにより, 遷移確率の推定を行なった.

例えば, ある局面から「直前に動いた敵の駒を駒得しながら取る手」によって別の局面に遷移する確率は, プロの棋譜において,「直前に動いた敵の駒を駒得しながら取る手」が可能である局面において, 実際にそのカテゴリの手が指された割合とした. ただし, 事例の数が少ないカテゴリや, 実戦棋譜からは抽出することのできないカテゴリの確率に関しては, ヒューリスティクスに基づく適当なスムージングにより算出を行なった.

表 1 に, 遷移確率の高い指し手のカテゴリのいくつかを, その確率値とともに示す. 範囲に幅がある物は, その指し手の仮評価 [5] によって確率値が異なることを示している.

ある指し手が複数のカテゴリに属する場合は, 最も高い確率のカテゴリを採用することとした.

2.3 再探索

今まで述べてきた実現確率打ち切りによる探索アルゴリズムを単純に実装すると大きな問題がおこる. それは, 確率の低い手による水平線効果である.

表 1: 指し手による遷移確率

カテゴリ	遷移確率
直前に動いた敵の駒を駒得しながら取る手	58%～89%
銀損以上の駒損になる駒が逃げる手	36%～69%
直前に動いた駒ではない駒を駒得しながら取る手	16%～67%
駒を取りながら王手を防ぐ手	51%
駒得をしながら王手をかける手	43%
駒損をしないで歩で王を叩く手	25%
駒損をしないで両取りをかける手	18%～40%
駒が駒損しないで成る手	13%～23%
:	:

実現確率打ち切りアルゴリズムでは，ノードの実現確率が閾値を下回った時点で末端になり評価関数が呼び出される．したがって，極端な場合，深さ 1 でもその指し手による遷移確率が非常に小さい場合は，そこで末端になってしまう．そうすると，少し不利な局面では，そのような非常に遷移確率の小さい手を選ぶことで，後の不利な変化を水平線の先に追いやってしまうことになる．

そこで本アルゴリズムでは，評価値を更新した手に対しては再探索を行なう．その場合に問題になるのは，再探索をする手の遷移確率をどのように設定するかということである．あくまでも，遷移確率という考え方に基づくのであれば，評価値を更新した手が最善手である確率を求めればよいのだが，それは再探索の確率をどう設定するかによって影響を受ける．すなわち，循環的な定義になっているため，実際に観測することは不可能である．そこで本論文では，再探索の遷移確率に関しては深く追求せず，0.5 とした．

図 2 に，再探索まで含めた実現確率打ち切りアルゴリズムを，C++ライクなコードで示す．

3 評価

3.1 実装

本探索アルゴリズムを実装した将棋プログラムを作成し，アルゴリズムの有効性の評価を行なった．
作成した将棋プログラムの特徴を以下に簡単に述べる．

- 指し手はカテゴリ毎に生成し，カテゴリと仮評価などの値に応じて確率値を付与する．また，残り探索確率が，その指し手が効果を発揮することを確認するのに不十分な場合，そのような指し手は生成しない．これは，YSS における指し手生成の考え方と同様である [5].
- 評価関数は，他の多くの将棋プログラムと同様に，駒の損得，駒の働き，玉の危険度などを主な評価項目とする関数である．
- 末端では単純に評価関数を呼び出すのではなく，静止探索を行なっている．静止探索とは，駒の取り合いなど，評価値が大きく変動する局面では，それが落ち着くところまで評価することで，正確な局面評価をしようとする手法である．本実装における静止探索は，単純な取り合いだけでなく，取る手，成る手，取る手や成る手を防ぐ手，も考慮して静止探索を行なっている．
- その他には，トランスポジションテーブルの利用や，類似ハッシュによる簡易必至探索などを行なっている．

```
int search(Shogiban* shogiban,
           double realization_probability,       // このノードの実現確率
           double min_realization_probability,   // 実現確率の閾値
           int alpha, int beta, Move& best_move)
{
  // 実現確率が閾値を下回ったら末端
  if (realization_probability < min_realization_probability)
    return leaf(shogiban);

  // 指し手生成（同時に，それぞれの指し手に遷移確率を付与）
  Move move[MAX_NUMBER_OF_MOVES];
  int num_moves = generate_moves(shogiban, move);

  int best_value = alpha;
  for (int i = 0; i < num_moves; i++) {

    // 指し手を進める
    shogiban->move(move[i]);

    // 最初の探索
    //   現局面の実現確率に指し手の遷移確率を掛けて再帰呼び出し
    //   （評価値を更新するかどうかを確かめるだけなので Null Window）
    Move dummy;
    int eval = -search(shogiban,
                       realization_probability * move[i].transition_probability,
                       min_realization_probability,
                       -(best_value+1), -best_value, dummy);

    if (eval > best_value) {
      // 評価値を更新した場合は遷移確率 0.5 で再探索
      eval = -search(shogiban,
                     realization_probability * 0.5,
                     min_realization_probability,
                     -beta, -best_value, dummy);
    }

    // 指し手を戻す
    shogiban->reverse();

    if (eval > best_value) {
      best_value = eval;
      best_move = move[i];
      if (best_value >= beta)
        return best_value;    // βカット
    }
  }
  return best_value;
}
```

図 2: 実現確率打ち切りによる探索アルゴリズム

表 2: 深さ打ち切りプログラムとの対戦

思考時間	勝	負	引き分け	勝率
一手 10 秒	177	21	2	0.894
一手 2 秒	116	82	2	0.586

(深さ打ち切りプログラムの思考時間は一手 10 秒)

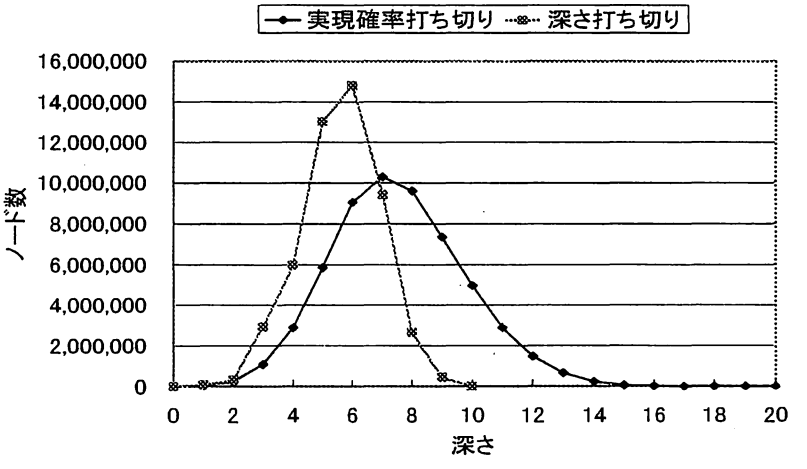

図 3: 探索ノードの深さ

実現確率による反復深化は，一回の反復深化で，実現確率の閾値を 0.25 倍していくことにより行なった.

3.2 深さ打ち切りアルゴリズムとの対戦による評価

最初に，棋力の評価を深さ打ち切りアルゴリズムとの対戦によって行なった.

本実験で用いた深さ打ち切りによるプログラムは，先に述べた確率打ち切りアルゴリズムの実装において，すべての指し手の確率を等しくすることで，深さ打ち切りアルゴリズムと同じ振舞をするようにしたものである.

勝率を調べるために多数の対戦を行なおうとするとき，初期局面から対戦を行なうと，ランダムな要素がない場合，毎回必ず同じ展開になってしまう．そこで本実験では，プロ棋士の棋譜から 21 手目の局面を多数取り出し，その局面から互いに先後一局ずつ戦わせることにより，異なった展開になるようにした.

表 3 に対戦結果を示す．実行環境は，PentiumIII 1GHz である．まず，両者ともに思考時間を一手 10 秒として対戦を行なった場合，実現確率打ち切りによる勝率は 9 割近くに達し，棋力に明確な差があることがわかる．そこで，確率打ち切りアルゴリズムの思考時間を一手 2 秒に減らして対戦を行なった結果，勝率は 6 割弱となった.

これらのことから，将棋プログラムの強さという点で，実現確率打ち切りアルゴリズムは，思考時間を等価的に 5 倍強にすることに相当することがわかる.

図 3 に，両者ともに一手 10 秒で対戦を行なった時の，探索ノードの深さを示す．ノード数は一局を通しての合計である．実現確率打ち切りによる手法では，深さ打ち切りと比べてかなり深いノードを探索して

表 3: コンピュータ将棋用次の一手問題集に対する正解数

実現確率打ち切りアルゴリズム		深さ打ち切りアルゴリズム	
総探索時間（秒）	正解数	総探索時間（秒）	正解数
133	24	157	20
346	23	419	20
802	25	1510	22
2103	31	5044	23

いることがわかる．

3.3 次の一手問題による評価

読みの正確さを評価するために,「次の一手問題集」による評価を行なった．評価に用いた問題集は，コンピュータ将棋の実力を測る目的で作成された標準問題集である [8]．実行環境は，PentiumIII 800MHz，ハッシュメモリ 128MB とした．表 3 に結果を示す．正解数で，実現確率打ち切りによる手法が，深さ打ち切りによる手法を大幅に上回っていることがわかる．また，この問題集に対する現時点での将棋ソフトの最高得点は，市販ソフトの「東大将棋 3」の 27 問であると報告されている [7]．総探索時間が同一ではない（東大将棋 3 は 1159 秒）ために，直接比較することはできないが，本アルゴリズムによる正解数の 31 問というのは，かなり高い正解率であるといえる．

また，上記の問題集以外にも，次の一手問題 100 問（雑誌「将棋世界」の昇段コース：初段・二段・三段コースの第 179 回～第 190 回，四段・五段コースの第 179 回～第 184 回，六段コースの第 179 回～第 185 回．詰将棋を除く）による評価を行なった結果，現在のトップレベルの将棋ソフトと比較して同等かそれ以上の正解率を達成することができた（表 4）．

表 4: 雑誌「将棋世界」に掲載の次の一手問題 100 問に対する正解数

プログラム	総探索時間（秒）	正解数
実現確率打ち切りアルゴリズム	1690	54
AI 将棋 2000（レベル 5）	1605	49
柿木将棋 5（レベル 6）	1638	47
東大将棋 3（マスターレベル）	2131	45

4 考察

前章の実験による結果，将棋プログラムの強さという点では，実現確率打ち切りアルゴリズムは大きな効果があることが確認された．

将棋の勝負に勝つためには，良い手を指すことよりもむしろ，極端に悪い手を指さないことの方が重要であるといわれている．その点，本アルゴリズムでは「普通」の展開を深く読むために，深さ打ち切りアルゴリズムでしばしば問題を引き起こす水平線効果が出にくくなり，それが結果として高い勝率に結びついていると考えられる．

また，次の一手問題に対しても本アルゴリズムは有効であった．一般に，次の一手問題では，駒をただで捨てる手など，一見「ありそうでない」手が正解であることが多い．本アルゴリズムは「ありそうな展開」を中心に読もうとするため，次の一手問題とは一見相性が悪そうに思われる．では，なぜ有効に働くのだろうか．以下の理由が考えられる．

一つは，次の一手問題は一見すると妙手を発見する力を問われているように見えるが，実際には妙手が妙手であることを見極めるために，その手に続く比較的平凡な手の展開を効率よく見定める力を問うていると思われることである．

もう一つは，いわゆる妙手は，その指し手の後に静的評価関数を一時的に悪化させるものが多い．このため，静的評価関数による評価を利用して探索窓を狭めるような枝刈り戦略においては発見できなくなる可能性が高い．実現確率打ち切りアルゴリズムは，この種の静的評価関数に基づく枝刈りは行なわず，しかも全体として資源配分を適正化できることが，次の一手問題の正解率を上げる原因になっていると考えられる．

5 まとめ

本論文では，局面の実現確率に基づくゲーム木探索アルゴリズムを提案し，そのアルゴリズムを実装した将棋プログラムによる評価を行なった．深さ打ち切りアルゴリズムとの対戦では，思考時間が同一の場合には9割近くの勝率を達成した．また，思考時間に5倍の差をつけても，6割近くの勝率となった．すなわち，本アルゴリズムは等価的に5倍強の探索速度向上に匹敵することを確認した．また，次の一手問題の正解数に関しても，深さ打ち切りアルゴリズムによる正解数を大きく上回り，他のトップレベルの将棋プログラムの結果と同等以上の正解数を達成した．

参考文献

[1] Hans J. Berliner and Chris McConnel. B* probability based search. *Technical Report CMU-CS-94-168, School of Computer Science, Carnegie Mellon University*, 1994.

[2] A. Junghanns. Are there practical alternatives to alpha-beta? *ICCA Journal*, Vol. 21, No. 1, pp. 14–32, 1998.

[3] D. McAllester and D Yuret. Alpha-beta-conspiracy search, 1993. url: http://www.research.att.com/~dmac/abc.ps.

[4] Aske Plaat, Jonathan Schaeffer, Wim Pijls, and Arie de Bruin. Best-first fixed-depth minimax algorithms. *Artificial Intelligence*, Vol. 87, No. 1–2, pp. 255–293, 1996.

[5] 山下宏. YSS-そのデータ構造、およびアルゴリズムについて. コンピュータ将棋の進歩2, pp. 112–142. 共立出版, 1998.

[6] 金沢伸一郎. 金沢将棋のアルゴリズム. コンピュータ将棋の進歩3, pp. 15–26. 共立出版, 2000.

[7] 松原仁. コンピュータ将棋の次の一手問題による評価. 情報処理学会研究報告 2001-GI-5, Vol. 2001, No. 28, pp. 39–46, 2001.

[8] 松原仁, 飯田弘之. 次の一手形式によるコンピュータ将棋の評価（その一） コンピュータ将棋の進歩2, pp. 61–111. 共立出版, 1998.

[9] 棚瀬寧. IS将棋のアルゴリズム. コンピュータ将棋の進歩3, pp. 1–14. 共立出版, 2000.