卒業論文

# ゲーム木探索で得られる形勢判断の信頼性の推定とその応用

平成 25 年 2 月 7 日提出

指導教員　近山 隆　教　授
鶴岡 慶雅准教授

電子情報工学科

03110412　亀甲 博貴

# 概要

人工知能の分野では，コンピュータに人間と同様の知性を持たせる試みの一環として知能ゲームをコンピュータにプレイさせる研究が盛んに行われている．その中でも特にオセロや将棋に代表される二人零和有限確定完全情報ゲームに関する研究は古くから行われており，オセロなどでは既に人間のトッププレイヤの実力を大きく上回るコンピュータプレイヤが存在する．将棋や囲碁など，より複雑なゲームでも，プログラムの実力が人間のトッププレイヤに近づいている．

将棋を始めとする多くのゲームのコンピュータプレイヤは，ゲーム木の探索を行うことでよりよい局面への到達を目指すアルゴリズムが主流である．この「よりよい局面」の判断は，評価関数と呼ばれる局面の形勢判断を行う関数を用いて行われる．しかし評価関数が返す評価値はあくまで注目する局面を点として見たときの優劣であり，それがどの程度安定しているかを表現しているものではない．その局面から優劣がどれだけ変化しうるのかを検討することは，その探索結果の信頼性を考える上で重要だと考えられる．

本研究では，浅い探索で得られる評価値と深い探索で得られる評価値の間の相関を調べた．またその結果から探索の信頼性を定義し，これを推定する手法を提案した．さらに，推定された信頼性を用いる応用手法として，ProbCut への応用と勝負手の生成手法を提案した．応用手法の評価は，それぞれ実際のプログラムに実装し，ゲームをプレイする中で行なった．

評価の結果，推定された信頼性の応用はそれぞれその有効性を示すには至らなかったものの，信頼性の推定には一定の精度が得られた．勝負手の生成手法は，通常探索では負けてしまう局面に対して行うことで，結果が変わる局面が存在することを確認した．

# 目次


1 はじめに 1

1.1 背景と目的 . . . . . 1

1.2 貢献 . . . . . 2

1.3 本稿の構成 . . . . . 2

2 関連研究 2

2.1 ゲーム木探索 . . . . . 2

2.2 ProbCut . . . . . 4

2.3 その他の枝刈り手法 . . . . . 6

2.4 局面毎にマージンを変化させる枝刈り手法 . . . . . 7

2.5 勝負手の生成 . . . . . 8

3 提案手法 9

3.1 深浅差の定義 . . . . . 10

3.2 深浅差の分布の探索への応用 . . . . . 10

3.3 深浅差の分布の推定モデルの構築 . . . . . 11

4 実験と評価 13

4.1 予備実験：深浅差の分布 . . . . . 13

4.2 分散の推定 . . . . . 15

4.3 ProbCut への応用 . . . . . 17

4.4 勝負手の生成 . . . . . 20

5 おわりに 22

5.1 まとめ . . . . . 22

5.2 今後の課題 . . . . . 22

謝辞 23

参考文献 24

# 1　はじめに

## 1.1　背景と目的

人工知能の分野では，コンピュータに人間と同様の知性を持たせる試みの一環として知能ゲームをコンピュータにプレイさせる研究が盛んに行われている．その中でも特にオセロや将棋に代表される二人零和有限確定完全情報ゲームに関する研究は古くから行われており，オセロなどでは既に人間のトッププレイヤの実力を大きく上回るコンピュータプレイヤが存在する [1]．一方で将棋や囲碁など，より複雑なゲームでは，コンピュータプレイヤは人間のトッププレイヤとの優劣がつけられていない，または実力の上で大きく離れている．しかし近年のコンピュータ将棋プログラムは人間のトップアマプレイヤやプロ棋士との対局でよい戦績を残しており，プログラムの棋力が人間のトッププレイヤに肉薄していることが分かる [2]．

二人零和有限確定完全情報ゲームは，その性質からコンピュータゲームプレイヤとの相性がよい．具体的には，多人数ゲームのようにプレイヤ間で協力関係を持つことでプレイングが変わる，または不確定ゲームのように偶発的なイベントで状況が変動するなどということがなく，純粋に先読みを行うことのみによって勝敗が分かれる．先読みは相手プレイヤの特性に左右されず，それ自体に深い知識を要するものでもないためアルゴリズムが書きやすい．

そのようなゲームの中で，本研究では将棋を対象とする．将棋を研究対象にあげる理由としては，その適度な複雑性がある．状態数を比較すると将棋はオセロより大きく囲碁より小さい．オセロは解かれていないゲームとはいえ既に人間の棋力を大幅に上回っている．一方で囲碁は将棋に比べて遙かに大きな状態数を持つ上に局面当たりの合法手や 1 ゲーム当たりの手数が大きく，また将棋の駒割のような分かりやすい損得基準が決めづらいことから評価関数が作りにくいため，評価関数と木探索を用いたアルゴリズムに適用し難い．そのため現在の主な囲碁プログラムはモンテカルロ法をベースとしたものが多い [3]．これらに対して，プログラムの棋力が向上する余地が十分にあるゲームであり，評価関数と木探索を用いたアルゴリズムが主流である将棋は，研究の対象として扱いやすいゲームであるといえる．

将棋プログラムの棋力が向上するアルゴリズム面での要因の 1 つに，探索効率の向上があげられる．ゲーム木探索を用いるゲームでは，一般にゲーム木を深く読む方がよい手を選択しやすいことが経験的に知られている．ゲームのルールとして各プレイヤには持ち時間が存在し，またマシンスペックにも限界がある．そのため，Bitboard の導入とその応用 [4] などといった，時間あたりの探索ノード数を増やす工夫とともに，限られた探索ノード数をより有効に使う工夫が棋力向上につながると考えられている．ゆえに探索する価値のないノードの探索を極力回避して指し手の選択を左右しうるノードをより多く探索する，Probcut [5]，Null Move Forward Pruning [6]，Futility Pruning [7] などの枝刈りの工夫はプログラムの棋力向上を図る上で欠かすことができない．

これらの工夫の一部は，浅い探索の結果から深い探索が不要であると予測できることに基づいている．これは浅い探索で得られる評価値と深い探索で得られる評価値の間には強い関係があり，深い探索で得られる評価値が取りうる範囲を予測することで可能となる．この関係は局面によって異なり，関係を正確に推定することでより効率的な枝刈りが実現可能となると考えられる．

本研究では，浅い探索で得られる評価値と深い探索で得られる評価値の間の関係を推定することを目的とする．

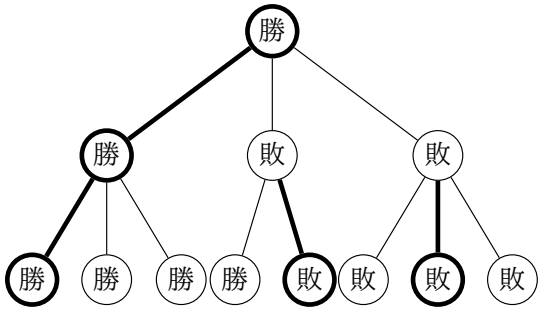


図 1. AND/OR ゲーム木

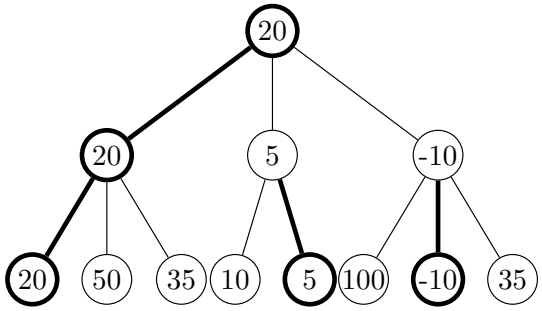


図 2. ミニマックス探索

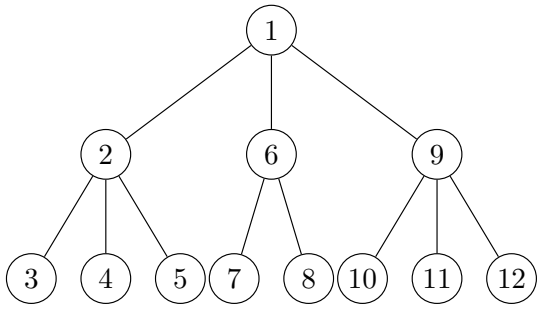


図 3. 深さ優先探索

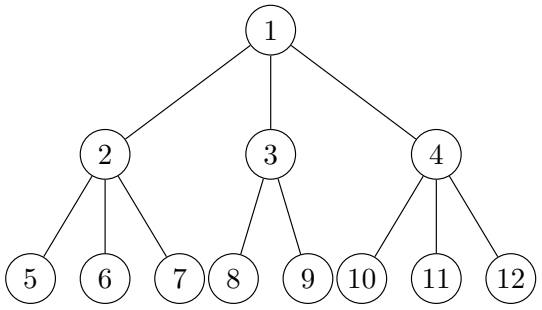


図 4. 幅優先探索

### 1.2 貢献

得られたモデルから推定された探索の信頼性は，探索に応用することでより効率的な探索を可能にすると考えられる．枝刈りの可否を判定する際に探索の信頼性を考慮することで，局面に応じた適切な枝刈りが期待できる．また，不利な局面においてはあえて不安定な局面を選択することで，形勢の逆転を目指すプレイングへと応用することができる．

### 1.3 本稿の構成

本稿は，第 2 章で関連研究の紹介，第 3 章で提案手法の説明を行う．第 4 章で行なった実験とその評価を行い，第 5 章で本研究のまとめと今後の課題を述べる．

## 2 関連研究

### 2.1 ゲーム木探索

図 1 のような，局面をノード・指し手をエッジで表した有向グラフをゲーム木という．リーフノードは終局面を表し，それぞれのノードが勝敗を持っている．引き分けのないゲームを仮定すると，手番プレイヤにとっての必勝局面が 1 つでも子ノードにあれば（OR），その局面は手番プレイヤにとっての必勝局面である．逆に全ての子ノードが手番プレイヤにとっての必敗手であれば（AND），その局面は手番プレイヤにとっての必敗局面である．このように，可能局面を全て展開することができるゲームであれば，AND/OR 木探索を行うことで，その局面での必勝プレイヤを決定することができる．

### 2.1.1 評価値と評価関数

Tic-Tac-Toe（三目並べ，○×ゲーム）のような小さなゲームであれば，初期局面からの可能局面を全て展開することが可能である．しかしオセロやチェス，将棋といった状態数の大きなゲームでは，時間計算量・空間計算量の両方の制約から不可能である．ゆえにある程度の深さで探索を打ち切り，勝敗に代わる指標を元に局面の選択をする必要がある．

局面選択の指標として主に用いられるのが，評価値とそれを返す評価関数である．評価関数とは局面に評価値と呼ばれる点数をつける関数で，図 2 のように，勝利に結びつきやすいリーフノードにはより高い点数をつける．局面の特徴に点数をつけ，それらを足しあわせたものを評価値とする評価関数が多く，例えばオセロでは，角を取ると高得点，その隣を取ると角が取られやすいから減点，といった点の付け方が考えられる．このアルゴリズムによるプレイヤの強さは用いる特徴に強く依存するので，用いる特徴とその点数の決め方が重要となる．局面選択の他の指標として，モンテカルロ法やその応用が用いられることもある．

### 2.1.2 深さ優先探索と幅優先探索

ゲーム木の探索の順序は，代表的なものが 2 つある．図 3 の順番での探索が深さ優先探索，図 4 の順番での探索が幅優先探索である．それぞれの探索には利点と欠点がある．深さ優先探索は幅優先探索と比較すると空間計算量が少ない点で勝るが，最短の終了ノードを見つける点では幅優先探索が勝る．これらを組み合わせることで欠点を補う，反復深化と呼ばれる探索もある．これは深さ優先探索を複数深さで行うもので，空間計算量が少ないままに浅いところにある終了ノードを見つけることができる．

### 2.1.3 ミニマックス探索と $\alpha\beta$ 探索

ゲーム木の探索は，自分はより自分に有利な手を，相手はより相手に有利な手，つまり自分に不利な手を指すことを目指すという前提で行う．このとき図 2 のようなゲーム木の最善ルートは自分の手番である奇数段目ノードでは子ノード中自身にとって最大評価のノードを，相手の手番である偶数段目ノードでは子ノード中自身にとって最低評価，相手にとって最大評価のノードを選択する．この探索アルゴリズムをミニマックス探索と呼ぶ．

全ての局面は AND/OR 木探索から得られる真の評価値としての勝敗を持つ．ある程度の深さで打ち切っての探索で得られる評価値は，この真の評価値の近似解であり，探索で得られた評価値が高い局面が常に勝利につながる局面ではない．一般に，より深く探索すると探索の精度が向上し，プレイヤが強くなることが経験的に知られている．ゲームのルールやハードウェアなどの制約から，探索を行える計算量は限られているため，その限られた計算量の中で効率の良い探索を行うことが，プレイヤの棋力向上につながるとされている．

図 5 のようなゲーム木の探索を考える．最左の部分木について探索を終えて次の部分木の最左経路を見た段階であり，次に探索すべきノードの評価値 $a$ は未知である．このとき図のノード $N$ の評価値は $x = min(10, a)$ となる．しかしこのときルートノードの評価値は $max(20, x, y) = max(20, min(10, a), y) = max(20, y)$ となり，$a$ の値によらずこの部分木が最大利益をもたらす経路ではない，そのためこのノードの評価値が指し手を左右することはなく，探索を行う必要がない．この場合は相手手番ノードの子ノードで評価値 20 以下のノードが現れた場合に，その部分木が枝刈りできることから，ノードが下限値 20 を持っていると考えることができる．これを $\alpha$ 値と呼ぶ．自分手番ノードの場合は逆に上限値を持ち，これを $\beta$ 値と呼ぶ．この $(\alpha, \beta)$ 窓をそれぞれ

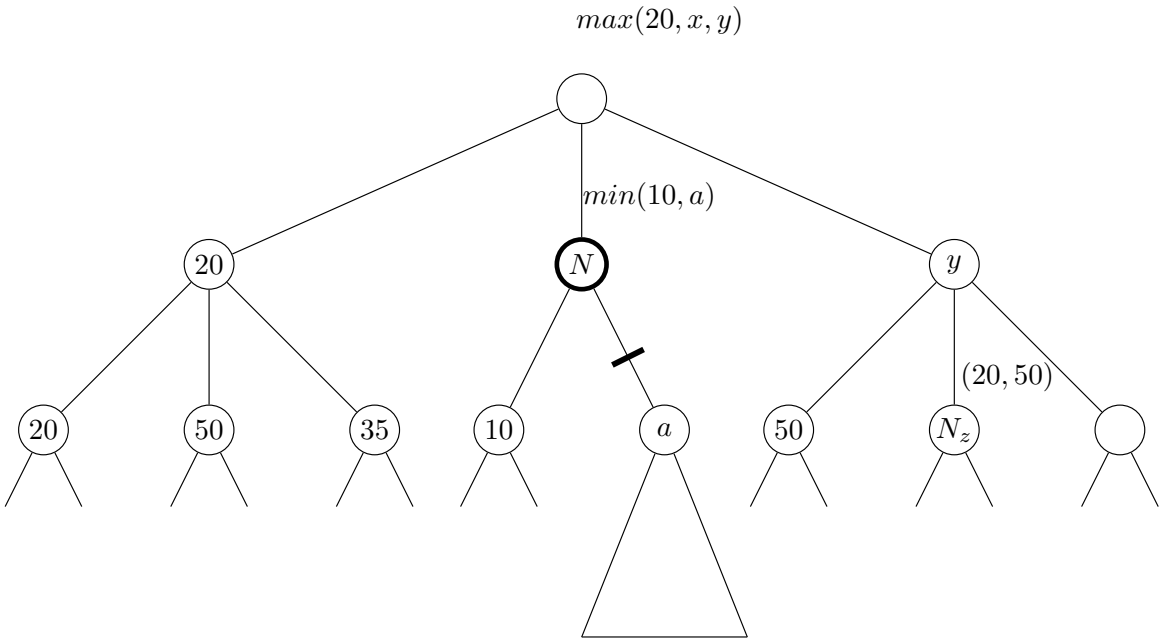


図 5. $\alpha\beta$ 探索

のノードが持ち，その外側の評価値が現れた箇所で枝刈りを行いながらの探索を $\alpha\beta$ 探索という．$\alpha\beta$ 探索は枝刈りを行なってもミニマックス探索と違いが生じない探索である．

このように，それぞれのノードは兄弟ノードや親ノードとの関係から，枝刈りされる評価値の範囲を持つ．例えば図 5 のノード $N_z$ は，最左ノードが評価値 50 のため，評価値 50 以上の場合はいくつであってもそのノードが最善手順となることはない．また，親ノードは兄弟ノードに評価値 20 のノードがあることから，ノード $N_z$ の評価値が 20 以下だと，親ノード毎このノードは枝刈りされる．つまり図の状態だとノード $N_z$ は評価値 20 から 50 までの範囲に無い限り最善手順ではない．このようなノードに対し，$(20, 50)$ という値の範囲を持つと考える．この値の範囲 $(\alpha, \beta)$ を $\alpha\beta$ 窓（Window）と呼ぶ．$\alpha$ は下限，$\beta$ は上限を表し，その外側の値は全て枝刈りされる．この $\alpha\beta$ 窓は兄弟ノードや親ノードから伝播する．そして $\alpha > \beta$ となったとき，枝刈りが発生する．

## 2.2 ProbCut

$\alpha\beta$ 探索は枝刈りが結果に影響を及ぼさない厳密な枝刈りである．このような枝刈りを後ろ向き枝刈りと呼ぶ．しかし後ろ向き枝刈りには限界があり，より多くの枝刈りを行おうとすると，$\alpha\beta$ 探索と同様の結果を返す厳密な枝刈りではなくなる．このような，枝刈りを行うことによって結果が変わりうる枝刈りを前向き枝刈りと呼ぶが，ProbCut [5] はそのような前向き枝刈りの 1 つで，浅い探索に高い信頼性があることを利用しての枝刈りである．

図 6 のような木を考える．図のノード N が残り深さ 8 であるが，一旦深さ 4 で探索を行う．その結果がノードの窓から著しく外れていた場合は，深さ 8 で探索を行なったとしてもおそらく枝刈りされるだろうという予測のもとで探索を打ち切る．枝刈りの可否の基準として一定のマージンを設定し，浅い探索の結果が $\alpha - margin$ より小さい場合は $\alpha$ カットを，$\beta + margin$ より大きい場合は $\beta$ カットを行う．

この枝刈りは，浅い探索から数手進めたときの評価値の変動量が一定値を下回ることを期待して

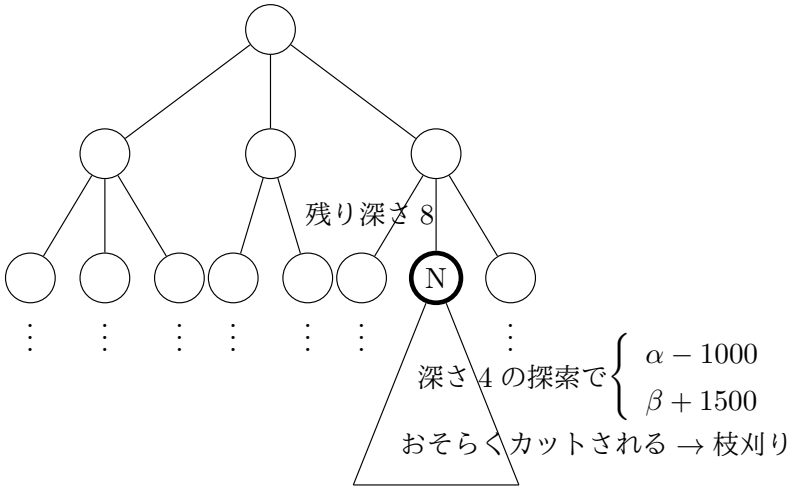


図 6. ProbCut

図 7. 浅い探索と深い探索で結果が大きく異なる局面 (shogipic.jp を用いて作成．)

いる．しかし，この変動量が予想される範囲を超えた値を取ると，誤った枝刈りを行なってしまいかねない．例えば図 7 のような局面を将棋プログラムの評価関数に与えると，深さ 4 での探索では評価値-271 であったものが深さ 8 での探索では評価値 383 となり，値が著しくずれてしまっている（歩 1 枚の基本価値は 100 である）．このような場合は ProbCut によって本来返すべきノードを枝刈りしてしまう危険性がある．しかし $\alpha\beta$ 探索自体が探索をある程度で打ち切る以上，特に中盤以前の局面では $\alpha\beta$ 探索が真に勝利へつながる手を返すとは限らない．そのため厳密に $\alpha\beta$ 探索と同じ結果であることを求める必要性は低く，ある程度結果が変わることについては許容できることが予想される．

ProbCut はオセロなどのゲームにおいてその有効性が示されている．将棋においても浅い探索の結果に高い信頼性が見られ，静止探索[*1]においてその有効性が示されている [8].

また ProbCut を複数深さ間で実行する Multi ProbCut という手法が考案されている．それぞ

[*1] 通常探索ではなく，駒の取り合いなどの手のみを読んで局面を静止させる探索

れの深さについて適切なパラメタを事前に用意することで，オセロで ProbCut や通常探索に十分勝ち越す結果を得ている [9].

### 2.2.1 Null Window 探索

ProbCut を行うにあたり枝刈りが可能であるかどうかの判定には浅い探索を行う．しかし枝刈りが生じない場合は深い探索も行わなければならないため二度手間となり，探索ノード数は逆に増えてしまう．浅い探索は，枝刈りが生じるか否かの判断さえできれば，それ以上の詳しい評価値を知る必要のないものである．浅い探索での余計な探索量を削減するための 1 つの手法として，Null Window 探索がある．

$\alpha\beta$ 探索はそのノードの評価値が (1) $\alpha$ 以下であるときは $\alpha$ カットされたという情報 (2) $\beta$ 以上であるときは $\beta$ カットされたという情報 (3) $\alpha < x < \beta$ となる値 $x$ であるときはその値 $x$, の 3 通りの結果を返す．(3) の場合以外は枝刈りが起こっており正確な値は判明しないが，それぞれ $\alpha$ 以下であるまたは $\beta$ 以上であるという情報を得ることはできる．そこであるノードの正確な評価値に興味がなく，評価値が $\alpha_0$ 以下であるかどうかのみを知りたい場合，$\alpha = \alpha_0$，$\beta = \alpha_0 + 1$ とするとその結果から $\alpha_0$ 以下であるか否かが判明する．このとき窓幅が 1 であるため，枝刈りが頻繁に生じる．Null Window 探索はこのように窓幅を 1 として $\alpha\beta$ 探索を行う手法で，早い段階で枝刈りが生じることから探索結果が注目する値以下であるか否かを高速で得ることができる．

## 2.3 その他の枝刈り手法

### 2.3.1 Null Move Forward Pruning

強力な前向き枝刈り手法の 1 つに Null Move Forward Pruning [6] がある．将棋はルール上パスができないが，仮にパスをしたとして探索を行う．適切な手を指した場合は，指さない場合より常によい状態を得られるという発想に基づいており，指さずにパスしたときの評価値が $\beta$ 値を超えていれば，適切な指し手を選べばより高い評価値を得られるだろうことが予想されるので枝刈りをするという手法である．

### 2.3.2 Futility Pruning

探索を行い，残り 1 手となったときにそのノードを試しに評価してみる．その評価値が $\alpha$ より一定以上低い，または $\beta$ より一定以上高いとき，後 1 手指してもその評価値が $\alpha$ と $\beta$ の間に入ることはないだろうという予測のもとで枝刈りをする，Futility Pruning [7] と呼ばれる手法がある．Futility Pruning はコンピュータ将棋選手権で優勝経験を持ち将棋プログラムに大きな影響を与えた Bonanza で応用され棋力の上昇が示されている [10].

### 2.3.3 実現確率探索

1 手ごとの深さを変動させる手法として実現確率探索 [11] が知られている．これはそれぞれの経路において各指し手が選ばれる確率（これを実現確率と呼ぶ）を利用した手法である．実現確率が高いということは，人が指すときに実際にそう指しやすいだろうということを意味する．例えば駒得でかつ王手をかける手があればその手を指す確率が高いことが統計的にわかっている（具体的には約 43%）ので，この場合の指し手の実現確率は高い．各経路の実現確率を掛けあわせていくとそのノードが実際に起こりうる確率が求まり，一定の基準値を下回ったところで探索を打ち切る手法である．例として図 8 をあげる．最左の経路の指し手の実現確率はそれぞれ 0.5，0.8 で，最左の

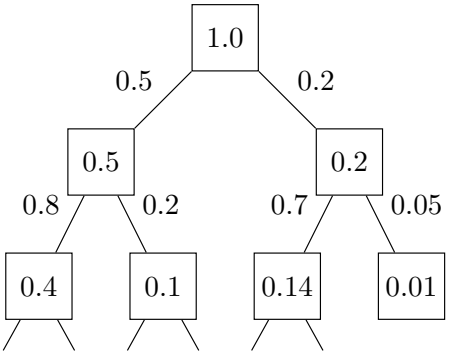


図 8. 実現確率探索

指し手をたどった局面の実現確率は $0.5 \times 0.8 = 0.4$ となる．一方で最右の指し手をたどった場合の実現確率は $0.2 \times 0.05 = 0.01$ となる．例えば実現確率 0.02 で探索を打ち切る場合，最左の経路はさらに展開するが，最右の経路はこれ以上展開せず，今の局面を評価して評価値を返す．結果として実現確率の高い経路についてはより深く，低い経路についてはより浅い探索で終えることになり探索木の形はあたかも枝刈りされたかのようになる．実装上は実現確率の対数を取りそれを足しあわせていくことで実現確率の掛けあわせを行なっている．この値が一定値を超えたところで探索を打ち切るので，一般的な木探索でいうところの深さが実現確率の対数の値に相当する．つまり各指し手を取ったときの残り深さの減少量が，実現確率の高い手では小さく低い手では大きくなる．以下で用いる「深さ」はこの実現確率の対数の値で表現する．但し平均的な指し手による深さの減少量は 1 であり，これは一般的な木探索でいうところの手数深さに相当する．

これらの枝刈りが ProbCut による枝刈り部分と共通する場合，浅い探索を不必要に行なってしまう可能性がある．特に Null Move Forward Pruning は ProbCut が起こるのと同じタイミングでも起こりうる枝刈りであるためその影響は大きいことが予想される．

## 2.4 局面毎にマージンを変化させる枝刈り手法

前項までで紹介した ProbCut，Null Move Forward Pruning，Futility Pruning といった枝刈り手法は，マージン値を小さくすると枝刈り量は多くなるが誤り枝刈りが発生する確率が高くなり，逆にマージン値を大きくすると誤り枝刈りが発生する確率は低くなるが枝刈り量は少なくなる．このように枝刈りの効率と安全性はトレードオフの関係にある．ゲーム全体を通して固定のマージン値を用いると，局面によっては枝刈りを発生させすぎて誤った枝刈りが多くなったり，逆に枝刈りが行われず効率の悪い枝刈りになってしまう危険性がある．そこで局面の状態によってマージン値を変動させる手法が考えられてきた．ここでは将棋においてマージン値を変動させた先行研究を紹介する．

通常探索に局面までの手数を用いてマージンを決定する ProbCut を適用し，その有効性を示した研究がある [12]．この研究において 2 手おきに評価値の差を取ることで統計的性質を調査し，初期局面から現在局面までの手数毎に標準偏差を計算するとそれが手数で直線に近似できることを示した．また前述の近似から求まる標準偏差に適切な係数を掛けたものをマージンとして用いた ProbCut が通常の $\alpha\beta$ 探索に有意に勝ち越すことを示した．

この研究に対し，各局面を序盤・中盤・終盤といった局面の展開に分類してマージンを定める手

法が提案された [13]．この研究ではゲームの展開を駒の衝突・駒の成り・寄せ・詰めといったイベントで分類し，各分類ごとに回帰を行なってマージンを決定する．また終盤は ProbCut が難しいとして適用範囲から外している．これら 2 つはいずれもマージンの決定関数を事前に与える手法である．

対して対局の途中でマージンを動的に変動させる手法を FutilityPruning に導入して棋力の向上を示した研究がある [14]．これは探索を行う中で仮に枝刈りを行なったとしてその枝刈りが正当なものである確率を実際に数え上げることで求め，探索中に一定数の統計が取れたらその確率を今探索している局面に用いて枝刈りを行う手法である．一定数の統計が取れるまでは枝刈りが行われないためある程度以上の計算時間を与える必要があるが，事前にプログラムにゲームに関する知識を与える必要がない点は前述の手法との大きな相違点である．

## 2.5 勝負手の生成

コンピュータプレイヤに勝負手を指させる研究はゲーム研究の 1 分野として行われており，将棋においてもいくつかの研究がある．しかし，探索の結果最善手として得られる手を指すのとは違い，勝負手にはいくつかの定義がある [15]．1 つは必死がかかっていたり詰められそうなとき，複雑な局面を選んで相手のミスを誘う手 [16] で，もう 1 つは不利な局面においてあえて最善手を避けて指すことで，不利な方向へ進んでいた流れの解消を目指す手 [17] である．前者を消極的な勝負手と呼ぶことがある．

相手のミスを誘う手を生成する場合を考える．自身は詰められることが分かっているため，相手が自身と同等の探索性能を持っていれば詰められてしまう．そこで，相手が自身より探索性能で劣っていることに期待し，相手が詰みを証明できないような局面を目指す．その方法として，詰み手数が長く，探索に要するノード数が多い手を優先したり，よく指されるために優先して探索される手であればそれを考慮したりといった要素から指し手を選択する．この手法で勝負手を生成するコンピュータプレイヤは，人間を相手にした対戦実験で一定の成果を上げた [16]．

一方で最善手を避けて指す手法は，相手の探索性能が劣っていることを仮定しない．例えば相手が自分より 1 手深い探索性能を持つ場合，図 9 と図 10 のように自分が探索する範囲は全て相手が探索してしまっているため，相手の探索結果を裏切る手を指すことができない．ゲームの展開が不利な方向に流れる原因は，このように相手の探索精度が自分より上回っていることによる場合が多い．このまま探索を続けると不利な流れのまま負けてしまいかねないので，相手の探索結果を裏切る手を指す必要がある．先行研究は，条件を満たす指し手について図 11 のようにより深く読み，一部分において相手の探索範囲に含まれない局面を読むことを目指した [17]．他の部分は相手の探索範囲より浅い探索になってしまい，探索精度は下がってしまうが，深く読んだ部分に関しては相手の探索を裏切る手を指す可能性があり，これを勝負手と定義した．

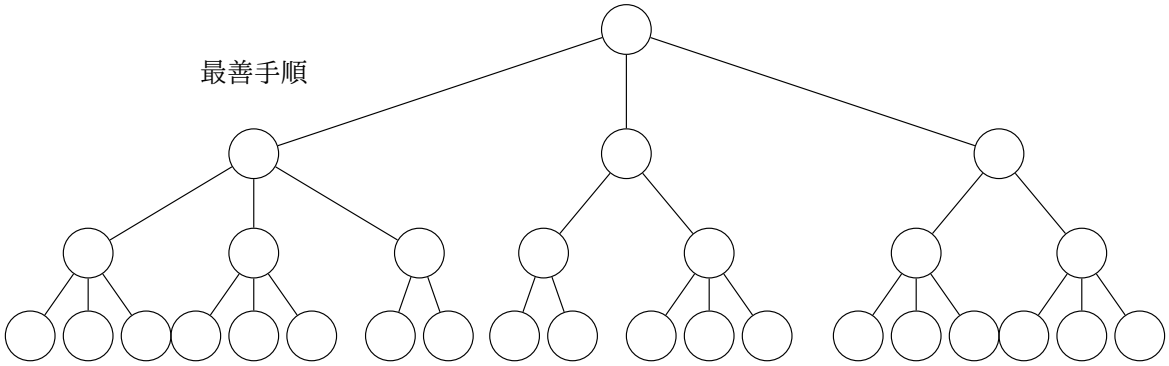


図 9. 相手の探索木

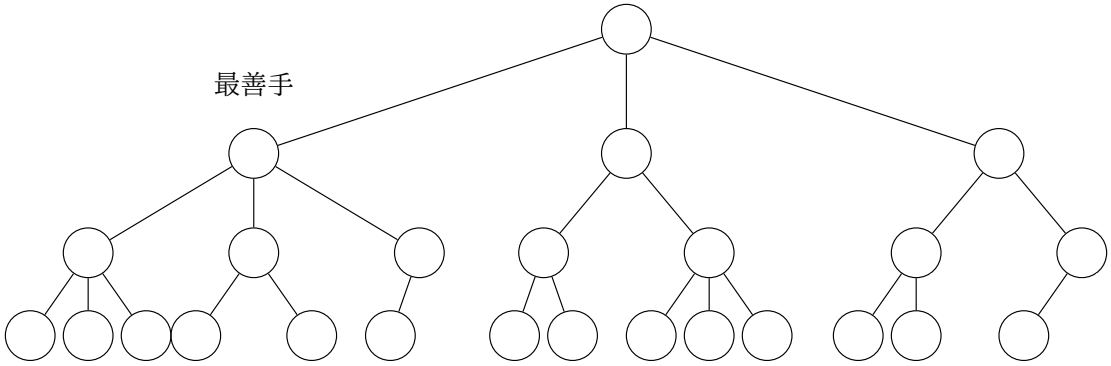


図 10. 自分の本来の探索木

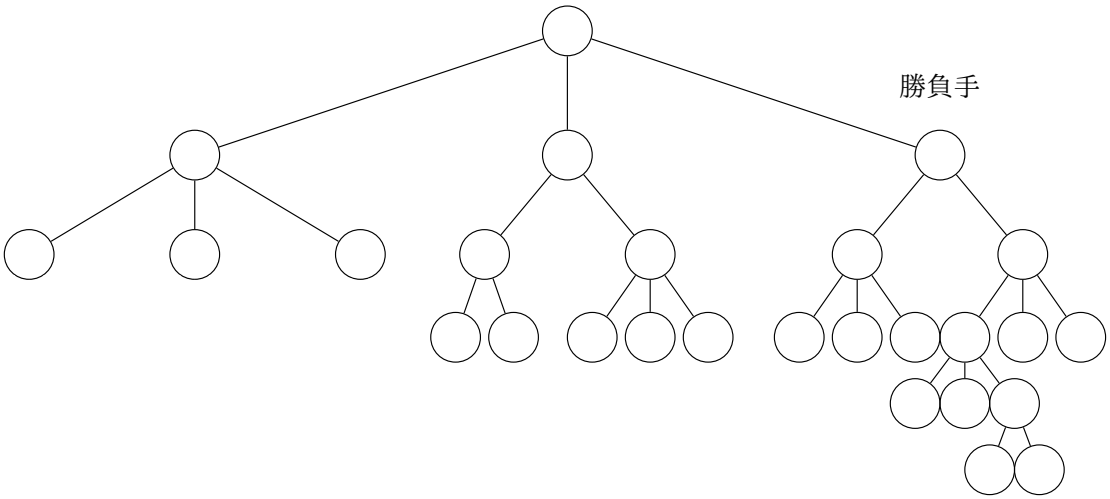


図 11. 相手の探索範囲を超える探索木

## 3　提案手法

探索により得られる評価値はその局面の真の評価値の近似解であり，深い探索を行うことでより精度の高い近似解が得られる．そこで本研究では，浅い探索の信頼性を，

浅い探索で得られた評価値が，深い探索を行うことによりどの程度変化するか

と定義する．浅い探索で得られた評価値と深い探索で得られた評価値の差が小さいなら，浅い探索の結果を深い探索の結果の近似解であると考えてよく，信頼性が高いといえる．逆に差が大きいな

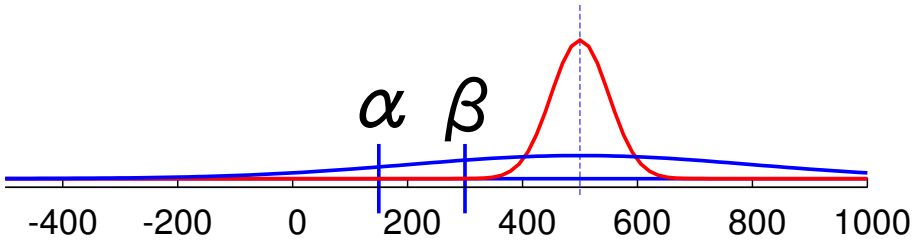


図 12. ProbCut による枝刈り発生判定の例

ら，浅い探索の結果から深い探索で得られる評価値がどの程度の大きさであるかを推測することが難しく，浅い探索の信頼性は低いといえる．

## 3.1 深浅差の定義

本研究では，浅い探索の信頼性を評価するにあたり，深浅差という概念を定義する．深浅差は

$$\text{深浅差} = (\text{深い探索で得られる評価値}) - (\text{浅い探索で得られる評価値})$$

で与えられるものである．

## 3.2 深浅差の分布の探索への応用

深浅差の分布を推定することで，様々な手法で探索への応用が期待できる．本研究では以下の 2 つの手法を提案する．

### 3.2.1 変動マージンによる ProbCut

ProbCut による枝刈りが生じるか否かの判定に，深浅差の分布を用いる．深浅差の分布と $\alpha\beta$ 窓が分かっていれば，累積確率を計算することで，そのノードの評価値が $\alpha\beta$ 窓に入る確率が求められる．この確率が十分に小さく，高確率で $\alpha$ カットまたは $\beta$ カットされるだろうと考えられるときは枝刈りを行う．

図 12 の場合を考える．$\alpha\beta$ 窓が $(150, 300)$ の局面で ProbCut による枝刈りが発生するかの判定を行う．横軸が評価値を表しており，点線は浅い探索で得られた評価値 500 を示す線である．赤い分布は浅い探索の信頼性が高い局面で，深い探索で得られるだろう評価値が浅い探索で得られた評価値と近いだろうことが期待される局面である．一方で青い分布は浅い探索の信頼性が低い局面で，深い探索で得られるだろう評価値が浅い探索で得られた評価値から大きく外れる可能性が十分にある局面である．このとき，赤い分布を取る局面で深い探索をして得られる評価値が $\alpha\beta$ 窓に入る確率は低く，青い分布を取る局面で深い探索をして得られる評価値が $\alpha\beta$ 窓に入る確率は無視できない程度には高い．このとき赤い分布の局面での枝刈り判定は比較的小さなマージンで，青い分布の局面での枝刈り判定は比較的大きなマージンで行うようにする．このように深浅差の分布に応じてマージンを変動させることで，局面に応じた適切な枝刈り判定が期待できる．

### 3.2.2 勝負手の生成

深浅差の分布を応用して勝負手の生成を試みる．ここでは「勝負手」を，

> 探索の結果得られる評価値は他の合法手に劣るが，探索を打ち切った先で評価値が 0 を上回

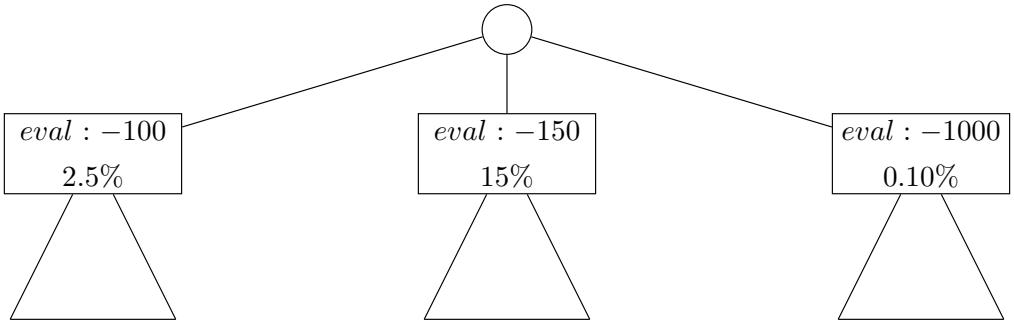


図 13. 勝負手生成

る可能性が他の手に比べて高い手

と定義する．

図 13 のような局面を考える．$eval$ は探索深さ $d_s$ の探索の結果得られた評価値，2 行目の確率は深い探索の結果評価値が 0 を超えるであろう確率である．この確率は浅い探索で得られる評価値と提案手法から計算されるものである．

通常の探索であれば評価値の一番高い手である最左を選択するが，この手を選んだときに評価値が正の値にまで回復する見込みは薄く，そのまま劣勢が続いてしまうと考えられる．一方で真ん中の手を選べば，探索の結果は劣るが探索を打ち切った先で評価値が正になる確率が 15% と，十分起こりうる確率である．同様の確率で評価値を大きく落としてしまう可能性もあるが，敗色濃厚な局面であれば指す価値が十分にある手を生成できることが期待される．

## 3.3 深浅差の分布の推定モデルの構築

### 3.3.1 深浅差の分布を与えるモデル

深浅差が母数$\boldsymbol{\theta}$と確率密度関数 $f$ で特徴づけられる確率分布 $P_X$ に従うと仮定する．確率密度関数 $f$ を事前に与えておけば，母数$\boldsymbol{\theta}$を推定することが深浅差の分布を推定することになる．

本研究では母数$\boldsymbol{\theta}$の各要素を

$$\theta_j = \boldsymbol{w_j}^T \phi(X_i)$$
$$\boldsymbol{w_j} \ : \text{重みベクトル}$$
$$X_i \ : \text{注目する局面}$$
$$\phi(X_i) \ : \text{局面 } X_i \text{ を表現する特徴ベクトル}$$

とするモデルを考える．

### 3.3.2 用いる特徴

先行研究 [12,13] が示す通り，深浅差の分布とゲームの展開は強い相関がある．局面の段階を示す指標の 1 つに，進行度というものがある．これは各駒の位置や王の危険度などによって求められ，値が大きいほど局面が終盤に差し掛かっていることを表現する．進行度は局面の段階を比較的低い計算コストで定義できる．

今回使用した将棋プログラム「激指」においては，

- 駒が自陣からどれだけ高い位置にあるか
- 駒が成っているか


図 14. 進行度の計算例

- 持ち駒であるか

の 3 つの項目で決定される．駒の種類に重み付けされており，その和を進行度とする．127 を超えた分については切り捨てられ，進行度は 0 から 127 の整数値をとる．図 14 の例では，

先手　1 三の歩兵：9
先手　1 五の香車：7
先手 持ち駒歩兵：$9 \times 3$
後手　5 五の歩兵：3
後手　8 九のと金：45

先手合計 43，後手合計 48 の平均をとって進行度 45 となる．

分布の母数$\boldsymbol{\theta}$の推定には，この進行度を決定する特徴を含む以下の特徴を用いる．

- 盤面上の駒の行位置と種類
- 持ち駒中の駒の種類と枚数
- 盤面上の駒がピンしている・されている[*2]
- 互いの玉の周りの駒
- 盤面上の駒の列位置と種類
- 互いの玉の周囲のマスに効いている駒の数は敵と味方でどちらがどれだけ多いか

盤面上の駒の行位置と種類・持ち駒中の駒の種類と枚数の 2 つは，進行度を決定する特徴である．

### 3.3.3　目的関数

モデルの作成は最尤推定により行う．最尤推定とは与えられたデータ群に対して最も尤もらしいモデルを推定する手法で，尤度の最大化問題である．

本研究では，尤度 $likelihood(\boldsymbol{w})$ は，分布の確率密度関数 $f(x)$ を用いて

$$likelihood(\boldsymbol{w}) = \prod_i f(x_i \mid \boldsymbol{w})$$

で計算される．高速化のため，実際には対数を取って $\log likelihood(\boldsymbol{w})$ の最大化を目指す．

[*2] 動くと玉将が取られてしまうため動けない駒をピンされている駒・動けなくさせている駒をピンしている駒という．

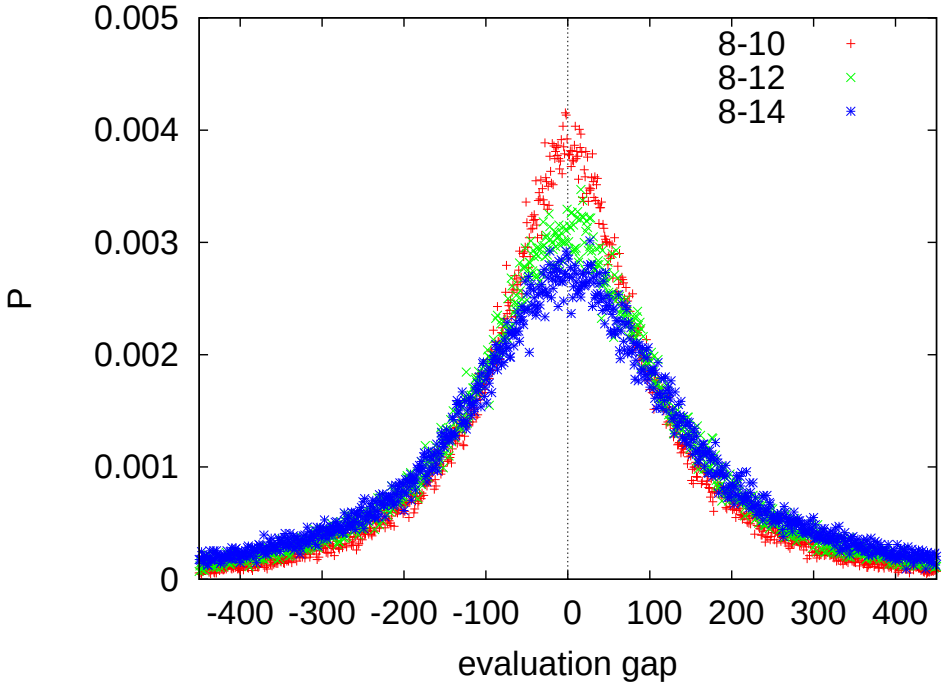


図 15. 浅い探索の深さ固定

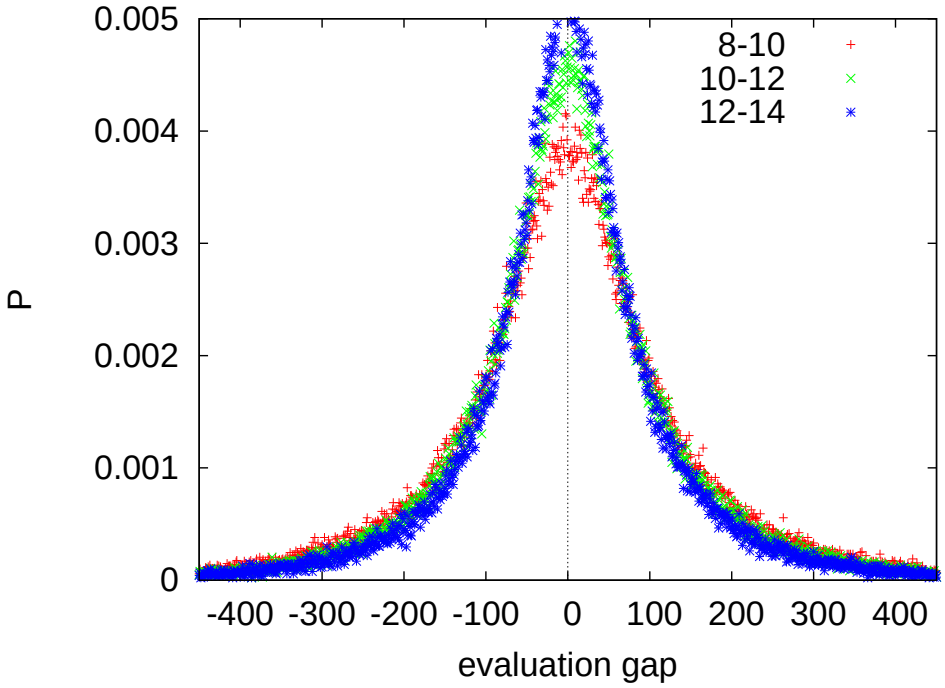


図 16. 浅い探索と深い探索の深さの差固定

# 4　実験と評価

## 4.1　予備実験：深浅差の分布

探索深さと深浅差の分布の関係を示したのが図 15 と図 16 である．図 15 は浅い探索の探索深さを 8 に固定し，深い探索の探索深さを 10，12，14 としたときの深浅差の分布，図 16 は浅い探索

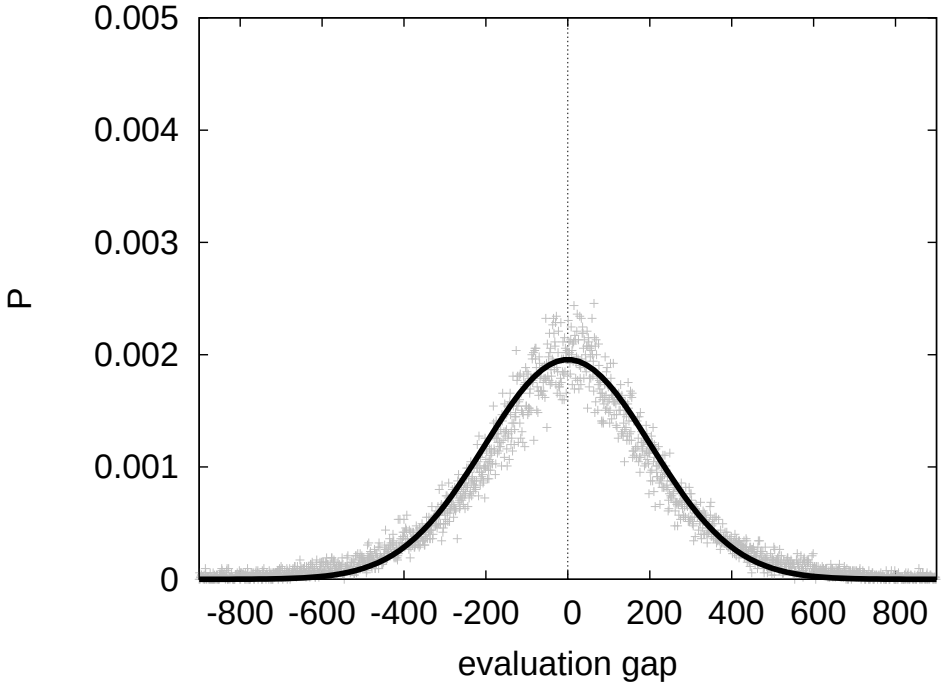


図 17. 深浅差の分布（進行度 48-63）

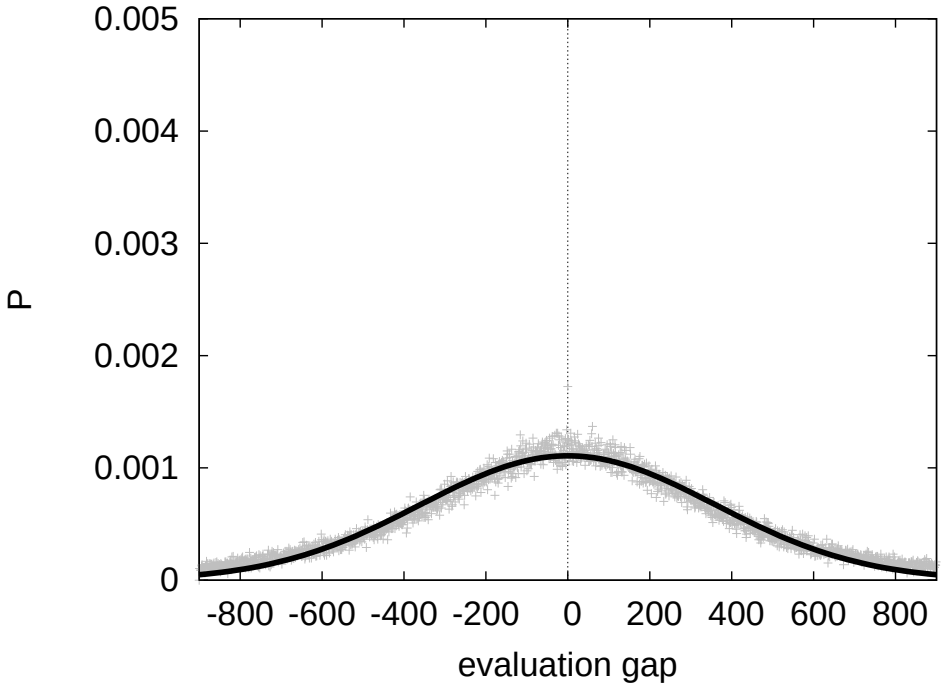


図 18. 深浅差の分布（進行度 112-127）

の探索深さと深い探索の探索深さの差を 2 に固定し，深さ 8 と深さ 10，深さ 10 と深さ 12，深さ 12 と深さ 14 の間の深浅差の分布を図示した．図 15 から，深浅差の分布は浅い探索と深い探索の間の差が大きいほど広くなり，図 16 から，浅い探索の探索深さと深い探索の探索深さの差が固定であれば，より深い探索の深浅差の分布が狭いことがわかる．

浅い探索深さ 4，深い探索深さ 8 の深浅差の分布をヒストグラムにしたのが図 17 と図 18 である．図 17 はルート局面の進行度が 48 から 63 のもの，図 18 はルート局面の進行度が 112 から

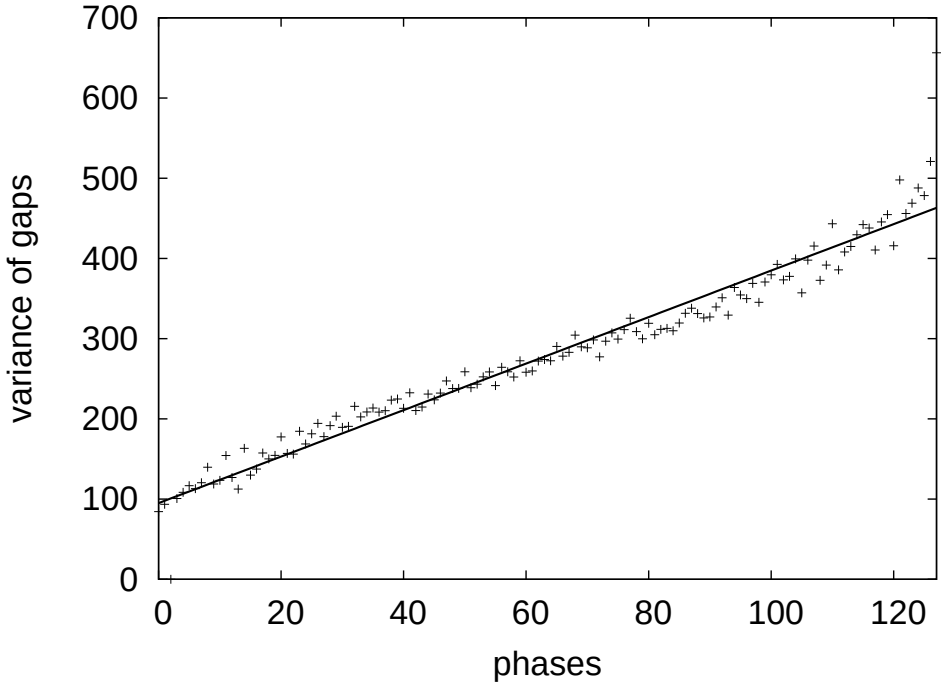


図 19. 進行度と標準偏差の関係

127 のものである．参考として正規分布を重ねている．それぞれ期待値 0 の正規分布様をしていることが分かる．

図 19 は各進行度における深浅差の標準偏差を計算したもので，深浅差の標準偏差と進行度の関係を示す．進行度が大きくなると正規分布の標準偏差が大きくなっている．進行度と標準偏差はほぼ一次の関係になっているといえる．

以上のことから，深浅差の分布として期待値 0 の正規分布を仮定し，その標準偏差を推定することができれば，浅い探索の結果から深い探索の結果が取りうる範囲を予測することができる．この標準偏差を信頼性の指標として用い，以下では標準偏差を推定することを目的とする．

## 4.2 分散の推定

### 4.2.1 目的関数

局面 $X_i$ は，深浅差 $g_i$ と特徴ベクトル $\phi(X_i)$ を持つ．このとき，深浅差の分布の標準偏差 $\sigma_i$ は

$$\sigma_i = \boldsymbol{w}^T \phi(X_i)$$

と表される．このときの深浅差の分布は，深い探索で得られる評価値 $\mu$，浅い探索で得られる評価値 $x$ として

$$\begin{aligned}\mathcal{N}(\mu, \sigma_i^2) &= \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{(\mu - x)^2}{2\sigma_i^2}\Big) \\ &= \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{g_i^2}{2\sigma_i^2}\Big)\end{aligned}$$

となる．このときの局面 $X_i$ の尤度を，図 20 のような期待値 $\mu =$ (深い探索で得られた評価値),

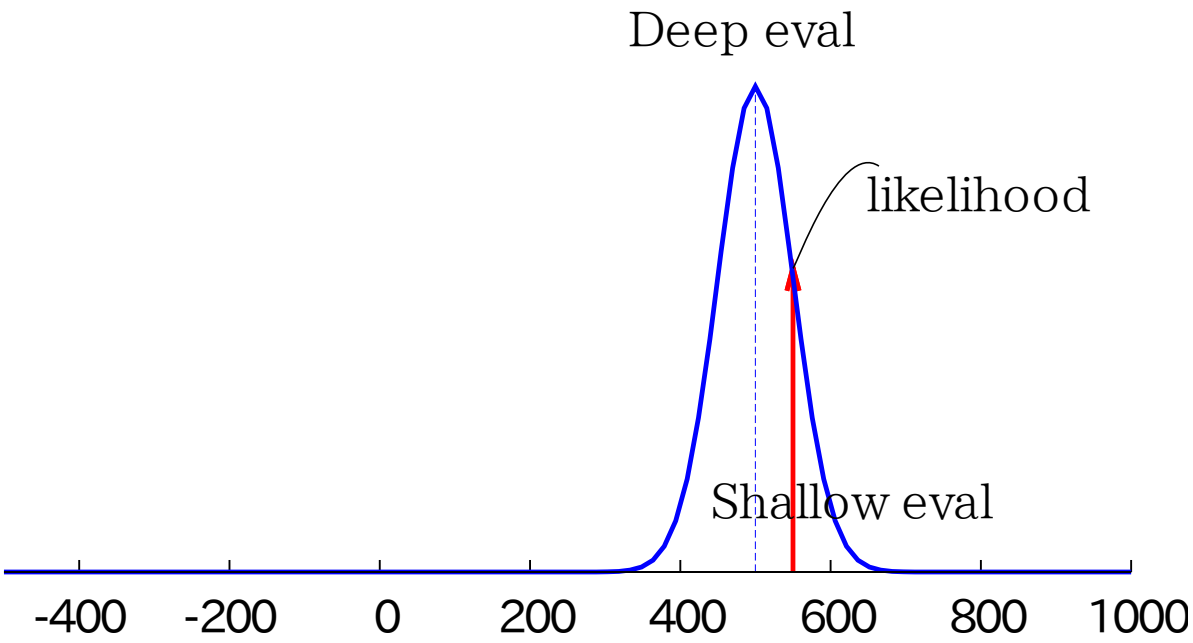


図 20. 尤度

標準偏差 $\sigma$ の正規分布を考えたとき, その分布に従う確率変数 $x$ が $x =$ (浅い探索で得られた評価値) となる確率と定義する．具体的な数式は

$$\frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{g_i^2}{2\sigma_i^2}\Big)$$

となる．全体の対数尤度は

$$\begin{aligned}
\log likelihood(\boldsymbol{w}) &= \log \prod_i \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\Big(-\frac{g_i^2}{2\sigma_i^2}\Big) \\
&= \log \prod_i \frac{1}{\sqrt{2\pi(\boldsymbol{w}^T\phi(X_i))^2}} \exp\Big(-\frac{g_i^2}{2(\boldsymbol{w}^T\phi(X_i))^2}\Big) \\
&= \sum_i \log \frac{1}{\sqrt{2\pi(\boldsymbol{w}^T\phi(X_i))^2}} \exp\Big(-\frac{g_i^2}{2(\boldsymbol{w}^T\phi(X_i))^2}\Big) \\
&= \sum_i \Big\{-\frac{g_i^2}{2(\boldsymbol{w}^T\phi(X_i))^2} - \log(\boldsymbol{w}^T\phi(X_i)) - \log\sqrt{2\pi}\Big\}
\end{aligned}$$

目的関数の勾配は

$$\begin{aligned}
\mathrm{grad}\log likelihood(\boldsymbol{w}) &= \frac{\partial \log likelihood(\boldsymbol{w})}{\partial \boldsymbol{w}} \\
&= \sum_i \frac{\partial}{\partial\sigma_i}\Big\{-\frac{g_i^2}{2\sigma_i^2} - \log\sigma_i - \log\sqrt{2\pi}\Big\}\frac{\partial\sigma_i}{\partial\boldsymbol{w}} \\
&= \sum_i \Big\{\frac{g_i^2}{\sigma_i^3} - \frac{1}{\sigma_i}\Big\}\phi(X_i)
\end{aligned}$$

本研究では最尤推定を行うのに libLBFGS[*3] という C ライブラリを用いた．これは FORTRAN

[*3] http://www.chokkan.org/software/liblbfgs/index.html

で書かれた準ニュートン法である L-BFGS (Limited-memory Broyden-Fletcher-Goldfarb-Shanno) [18, 19] を C で書きなおしたものである．

#### 4.2.2 特徴を増やすことによる尤度の変化

それぞれの特徴と尤度の関係を表 1 に示す．なお，尤度はそれぞれ対数尤度で，示した数字は尤度の平均である．訓練データは 642,249 局面，開発データは 160,574 局面で，全て同一のものを使用した．特徴を増やすことで尤度が上昇していることが分かる．

#### 4.2.3 推定の精度

提案手法がどの程度標準偏差を推定できているかを確認した．各局面について提案手法を適用し，(深浅差 $g_i$, 推定された標準偏差$\sigma_i$) の組を作った．これを $\sigma_i$ で整列し，128 等分した．それぞれのデータ群 128 個について，推定された標準偏差の平均 $\sigma_{0j} = \overline{\sigma_i}$ と実際の標準偏差 $\sigma_j$ を求めた．横軸に $\sigma_{0j}$ を，縦軸に $\sigma_j$ をプロットしたものが図 21，図 22 である．エラーバーは推定された標準偏差の標準偏差を表している．緑の直線は $x = y$ で，赤い点がこの直線に近いほどよく標準偏差を推定できていると言える．

図を見ると，標準偏差の小さい部分では両方ともよく推定できているように見える．しかし標準偏差が大きくなると，進行度の要素のみを用いたものは推定の精度が落ちていることが分かる．一方で，提案した特徴全てを用いたものは，いずれの局面においてもよく推定できているように見える．両方の図で共通して，標準偏差が最大のデータ群ではエラーバーが大きくなっている．これは推定された標準偏差が大きいところではその分布に属する局面が少なく，推定された標準偏差がデータ群の中で広い範囲を取ることを意味している．つまり一定の範囲内の標準偏差に対する局面数が少なく，教師データが相対的に少なくなる．標準偏差の小さいところがよりよく推定できているのはこのためであると考えられる．

### 4.3 ProbCut への応用

#### 4.3.1 ProbCut の実装

本研究で用いた ProbCut の実装を以下に記す．

探索中に表れたノードのうち，残り深さが 8 以上 11 未満のものを対象とする．また，$\alpha$ 値が $-30,000$ 未満のノードは対象外とする．このノードの特徴を用いて，提案手法により標準偏差 $\sigma$ を計算し，適切な係数 $n$ をかけて $n\sigma$ をマージンとする．$(\alpha - n\sigma, \alpha - n\sigma + 1)$ で，残り深さ $-4$ の Null Window 探索を行う．これが $\alpha - n\sigma$ を返したとき，詰み探索を行なって詰みが見つからなければ枝刈り．

枝刈り判定の対象が深さ 8 以上 11 未満であるが，マージンの決定に用いるモデルは，浅い探索

表 1. 用いた特徴と尤度の関係

| 用いた特徴 | 学習率 | 訓練データの尤度 | 開発データの尤度 |
|---|---|---|---|
| 進行度 | 0.005 | -7.367 | -7.363 |
| 進行度要素 | 0.005 | -7.074 | -7.071 |
| 提案手法 | 0.005 | -7.054 | -7.053 |

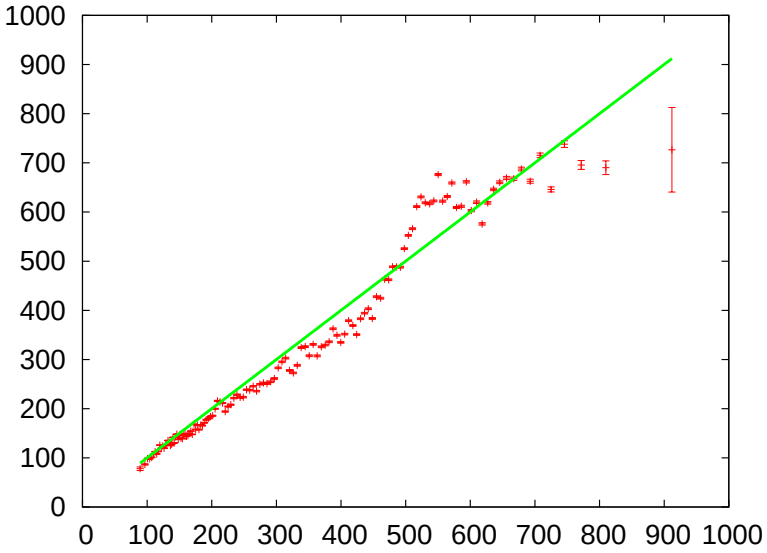


図 21. 進行度の要素

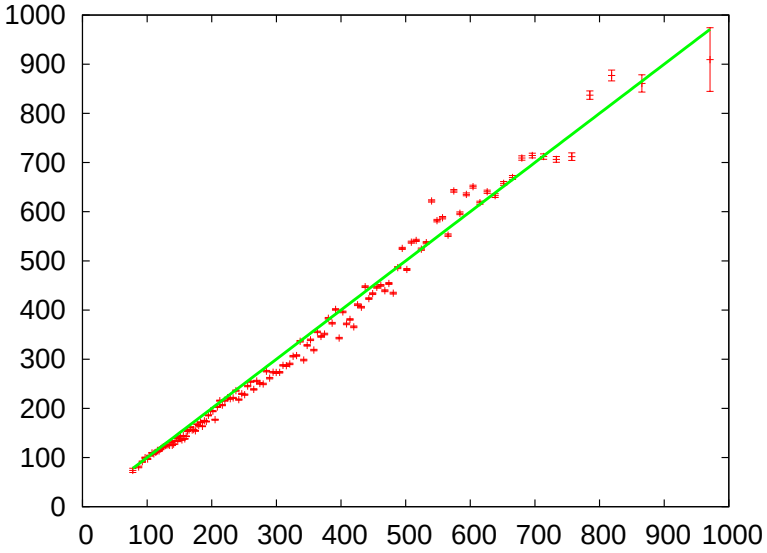


図 22. 提案した特徴

の探索深さ 4，深い探索の探索深さ 8 の深浅差から学習されたものを用いる．これは前項から，4-8 間の深浅差の分布を用いて枝刈り判定を行えば安全であるためである．そのため実際の深浅差の分布より広い分布を用いての枝刈り判定を行うことになる．

対象深さが 8 以上 11 未満と幅があるため，擬似的な MultiProbCut を行う実装となっている．また，事前の調査から $\beta$ 側の枝刈りは行わなくても枝刈り効率に大きな差が現れないため，$\alpha$ 側のみの枝刈り判定を行う．

以降の調査では，激指に ProbCut を実装しての実験を行なった．「ProbCut 無し」とは ProbCut を一切行わないもの，「進行度のみ」とはマージンを $200 + (\text{進行度} * 4)$ としたもの，「提案手法 $(n\sigma)$」とは，提案手法により得られた標準偏差 $\sigma$ に $n$ をかけたものをマージンとしたものを指す．他の設定は全て共通である．

#### 4.3.2 ProbCut による探索ノード数の削減量

ProbCut が探索ノード数をどの程度削減するか，探索深さを固定しての探索で比較した．プロの棋譜中から抜き出した 1,000 局面について探索を行い，プロの手との一致率・探索にかかった時間・ノード数を比較した．プロの手が必ずしも必勝手とは限らず，一致率が高いことが棋力の高さを意味するわけではないが，一致率を落とさずに探索ノード数を削減することができれば，比較的安全に枝刈りが行えていると考えてよい．結果を表 2 に示す．

結果から，ProbCut を行うことで探索ノード数を削減できていることがわかる．正解数にそれほどの変化が無く，この結果を見る限りでは，棋力を落とすことなく計算量の削減に成功している．

#### 4.3.3 ProbCut を行うことによる棋力の向上

棋力の比較は対戦実験により行う．対戦の設定は

- 棋譜を 30 手読み，これを開始局面とする
- 開始局面から先後を入れ替えて 2 対戦行う
- 2 対戦で共に勝敗のついた局面のみを有効な局面とする
- 1 手の時間を固定する

とする．なお，1,000 対戦での 5% 有意水準は勝率 51.6% である．結果を表 3 に示す．

ProbCut を行わないものに対して提案手法は有効に働き，有意に勝ち越した．しかし進行度のみを用いて ProbCut を行なったプレイヤに対しては有意な棋力差が見られず，提案手法の有効性を示すには至らなかった．

ProbCut へ提案手法を適用することで有意な棋力差が見られなかった原因として，その枝刈り性能に十分な差が見られないことが挙げられる．前項の実験結果から，進行度のみを用いての ProbCut と提案手法 $1\sigma$ とでは，探索時間にして 3.2% の削減が見られる．しかし今回使用したプログラム「激指」において，探索深さを 1 増やすと，探索時間は約 2.5〜3.5 倍になる．探索時間

表 2. 探索ノード数の削減量比較

| 手法 | 深さ | 正解数 | 時間（秒） | ノード数 | ノード数/秒 |
|---|---|---|---|---|---|
| ProbCut 無し | 13 | 574 | 1,427 | 399M | 279,978 |
| 進行度のみ | 13 | 569 | 1,285 | 361M | 280,957 |
| 提案手法（$1\sigma$） | 13 | 570 | 1,244 | 346M | 277,973 |
| 提案手法（$2\sigma$） | 13 | 574 | 1,364 | 385M | 281,966 |
| 提案手法（$3\sigma$） | 13 | 572 | 1,411 | 399M | 282,902 |

表 3. 対戦実験の結果

| 対戦プレイヤ | | | 1 手 | 結果 | 勝率 |
|---|---|---|---|---|---|
| 提案手法（$3\sigma$） | vs. | ProbCut 無し | 3s | 478-426 | 0.529 |
| 提案手法（$1\sigma$） | vs. | 進行度のみ | 3s | 439-457 | 0.490 |
| 提案手法（$2\sigma$） | vs. | 進行度のみ | 3s | 449-443 | 0.503 |
| 提案手法（$3\sigma$） | vs. | 進行度のみ | 3s | 441-481 | 0.478 |

を 3.2% 削減したことで探索範囲が劇的に増えることはなく，その効果が棋力としては現れにくいことが原因であると考えられる．一方で提案手法 $2\sigma$ では，探索時間は進行度のみのものに比べて多いものの，$1\sigma$ より対戦結果が多少改善した．ProbCut のマージンにはある程度の余裕を持たせて，安全な枝刈りを行うことが必要であると考えられる．

## 4.4 勝負手の生成

### 4.4.1 勝負手の生成手法

勝負手の生成のために，ルートノードの子ノードをそれぞれ部分木のルートノードとし，通常探索を行う．それぞれの部分木について評価値が 0 を超える確率の一番大きいものを選択する．ただし評価値の絶対値が $3\sigma$ を超えるとその確率を正確に計算することができず，確率 0 または 1 となってしまうため，評価値そのものを用いる．

### 4.4.2 勝負手の生成実験

手法の評価は，自己対戦により行う．棋譜を一定手数読み，その局面から通常探索をした場合と勝負手生成探索をした場合で 2 局面作成する．その後勝負がつくまで通常探索を行い，勝敗が変動するかを見る．

1 手目の通常探索と勝負手生成探索は深さ 12 で行い，以降の自己対戦は深さ 6 で行なった．200 手（お互いに 100 手）指して決着がつかなかった局面は引き分けとし，結果には含めなかった．結果を表 4 に示す．

勝負手を生成することで負けが勝ちになる局面があることは確認できたが，ほぼ同数の局面で本来勝ちの局面で負けてしまった．初期手数 79 手では，結果が変わらなかった局面数が 479 局面あったのに対し，343 局面では結果が変わった．また，そのうち 155 手は本来は負けていた局面で結果が勝ちに変わっており，通常探索を行うよりよい結果につながる局面が多く存在した．一方で本来はそのまま続けていれば勝っていた局面で，勝負手を指したために負けてしまった局面が 188 局面あった．開始局面の静的評価値毎にその数を調べたが，いずれの評価値においても負けが勝ち

表 4. 勝負手生成実験

| 初期手数 | 初期局面の評価値 | 敗け → 勝ち | 勝ち → 敗け |
|---|---|---|---|
| 79 | ～−1000 | 5 | 3 |
| 79 | −1000～−500 | 16 | 14 |
| 79 | −500～0 | 33 | 40 |
| 79 | 0～500 | 50 | 67 |
| 79 | 500～1000 | 36 | 31 |
| 79 | 1000～ | 15 | 33 |
| 119 | ～−1000 | 0 | 1 |
| 119 | −1000～−500 | 7 | 6 |
| 119 | −500～0 | 10 | 8 |
| 119 | 0～500 | 6 | 8 |
| 119 | 500～1000 | 9 | 9 |
| 119 | 1000～ | 12 | 6 |

になる局面の数が勝ちが負けになる局面の数を大きく上回っているわけではなく，評価値によって勝負手の生成を行うかどうかの決定はできないことがわかった．

初期手数 119 手では評価値 1000 以上の，先手有利な局面で後手が勝負手を生成したときは，有意な数とはいえないものの負けが勝ちになる局面の数が多かった．勝敗が変わった局面が 82 に対し，勝敗が変わらなかった局面が 3 倍の 230 あり，初期手数 119 手では勝負手を生成することで結果があまり変動しないことがわかった．

本研究で提案した勝負手生成は，適用すべき局面の判別ができていないため，実際の対戦において有効であるとはいえないが，結果が有利な方向に変わる手が生成できている局面が多く見られ，適用すべき局面の判別が可能であれば有効な手法になることが期待される．

# 5　おわりに

## 5.1　まとめ

本論文では，深浅差の概念を定義し，探索結果の信頼性の指標に深浅差の標準偏差を用いた．深浅差の標準偏差を探索に応用する手法として，ProbCut と勝負手の 2 つを提案した．

標準偏差の推定は，最尤推定により行なった．推定された標準偏差と実際の標準偏差を図示することで，提案手法が標準偏差の推定をよく行えていることを確認した．

探索深さを固定した一致数の比較実験では，棋力の低下が見られず，探索ノード数の削減が確認できた．対戦実験では，ProbCut を行わないプレイヤには有意に勝ち越したが，進行度のみを用いてマージンを決定する ProbCut を行うプレイヤに対しては，いずれの設定においても有意には勝ち越さず，棋力の向上が見られなかった．

勝負手の生成実験では，誤って負けにつながる手を指してしまうことも多かったが，通常探索を行うと負けてしまう局面において，適用することで勝ちにつながる局面が多く存在することが確認された．生成の適用範囲を適切に設定することで，棋力向上に有効であることが期待できる．

## 5.2　今後の課題

ProbCut を行うにあたり，その実装は

- 対象ノードへ到達
- マージンの決定
- Null Window 探索
- 枝刈りの決定

の順番で行われる．そのため探索結果をマージンの決定に反映させることができず，対象ノードの特徴のみから決定せざるを得なかった．例えば探索木の形などといった，探索の結果リーフから返ってくる特徴が標準偏差に影響する可能性は十分に考えられるので，探索結果を考慮した標準偏差推定モデルの構築を考える必要がある．

本研究での ProbCut の実装は，枝刈り判定には浅い探索の探索深さ 4，深い探索の探索深さ 8 の深浅差の分布モデルを用いて，対象ノードは残り深さ 8 以上 11 未満としてのものである．しかしこの実装だと，多くの場合浅い探索の探索深さ 7，深い探索の探索深さ 11 に近いノードでの枝刈り判定を行なっている．そのため本来適用するべき分布モデルとは大きく異なるものを用いてしまっている可能性がある．本来 MultProbCut を行う際は，各深さ毎に枝刈り判定のモデルを作成する必要があり，そうすることで棋力の向上が期待できる．その際 ProbCut 判定を行う深さなどのパラメタを調整し，より ProbCut が有効に働く設定を調査する必要がある．

勝負手生成によって，敗けが勝ちになる局面があることはわかったが，ほぼ同数の局面で勝ちが敗けになる局面も存在した．これらの局面を判別し，有効な局面でのみ勝負手生成を行うようにする必要がある．本研究では注目局面の評価値を見たが，これからは判別ができず，他の指標から判別を行う必要があると考えられる．また，本研究では評価値が正になる確率を用いたが，これは 1 手の勝負手で逆転を目指すものであり，大勢の決した局面での適用は難しいと考えられる．確率を計算する基準の評価値を $\alpha$ 値にするなどといった工夫により，評価値が負ではありながら有利な方向に進む手の選択をするという手法が考えられる．

# 謝辞

本研究を進めるにあたって，多くの方にお世話になりました．

指導教員である近山隆教授には，研究の方向性に始まり，様々な面でアドバイスをいただきました．また，研究に対する心構えを教えてくださいました．

同じく指導教員である鶴岡慶雅准教授には，研究の内容や進め方について数多くのアドバイスをいただきました．

マンチェスター大学の三輪誠さんには，技術的なアドバイスなど細部にわたって助けていただきました．

研究室博士 2 年の浦晃さんには，実験に使用するデータを作っていただき，また研究について多くのアドバイスをいただきました．

研究室の皆様には，研究・生活の両面でお世話になりました．本研究を進めるにあたっても，多大なご支援をいただきました．

この場を借りて厚くお礼申し上げます．

# 参考文献


[1] Buro, M.: Takeshi Murakami vs. Logistello, *ICCA Journal*, Vol. 20, No. 3, pp. 189–193 (1997).

[2] 勝又清和: 清水市代 VS あから 2010, コンピュータ将棋協会誌, Vol. 22, No. 12, pp. 61–64 (2010).

[3] Gelly, S., Wang, Y., Munos, R. and Teytaud, O.: Modification of UCT with Patterns in Monte-Carlo Go, Rapport de recherche RR-6062, INRIA (2006).

[4] Heinz, E.: How DarkThought Plays Chess, *ICCA Journal*, Vol. 20, No. 3, pp. 166–176 (1997).

[5] Buro, M.: ProbCut: An Effective Selective Extension of the alphabeta Algorithm, *ICCA Journal*, Vol. 18, No. 2, pp. 71–76 (1995).

[6] Donninger, C.: Null Move and Deep Search: Selective Search Heuristics for Obtuse Chess Programs, *ICCA Journal*, Vol. 16, No. 3, pp. 137–143 (1993).

[7] Heinz, E.: Extended futility pruning, *ICCA Journal*, Vol. 21, No. 2, pp. 75–83 (1998).

[8] 竹内聖悟, 金子知適, 川合慧: 将棋における ProbCut の静止探索への応用, 情報処理学会研究報告, Vol. 2005, No. 87, pp. 9–15 (2005).

[9] Buro, M.: Experiments with Multi-ProbCut and a new high-quality evaluation function for Othello, *Games in AI Research*, pp. 77–96 (1997).

[10] 保木邦仁: コンピュータ将棋の新しい動き : 3. コンピュータ将棋における全幅探索と futility pruning の応用, 情報処理, Vol. 47, No. 8, pp. 884–889 (2006).

[11] Tsuruoka, Y., Yokoyama, D. and Chikayama, T.: Game-Tree Search Algorithm Based on Realization Probability, *ICGA Journal*, Vol. 25, No. 3, pp. 145–152 (2002).

[12] 吉原一期, 近山隆: ProbCut の改良と将棋への適用, コンピュータソフトウェア, Vol. 19, No. 3, pp. 201–204 (2002).

[13] 柴原一友, 乾伸雄, 小谷善行: ProbCut の適用有効性 - 局面の分類ごとにパラメータを変えた場合 -, *The 7th Game Programming Workshop (GPW 2002)*, pp. 73–80 (2002).

[14] 伊藤裕, 橋本剛, 橋本隼一: 動的なマージンを用いる Futility Pruning, *The 12th Game Programming Workshow (GPW 2007)*, pp. 1–8 (2007).

[15] 松原仁, 飯田弘之: ゲーム・プレイングにおける勝負手, 情報処理学会研究報告. 人工知能研究会報告, Vol. 95, No. 23, pp. 111–118 (1995).

[16] Kajihara, Y., Hashimoto, T. and Iida, H.: A Speculative Play in Shogi Endgame, *The 7th Game Programming Workshop (GPW 2002)*, pp. 57–64 (2002).

[17] OTSUKI, T. and AIHARA, K.: Game-Tree Search with Flexible Control and Its Application to Generate'Syoubu-te'., *The 7th Game Programming Workshop (GPW 2002)*, Vol. 2002, No. 17, pp. 51–56 (2002).

[18] Nocedal, J.: Updating quasi-Newton matrices with limited storage, *Mathematics of computation*, Vol. 35, No. 151, pp. 773–782 (1980).

[19] Liu, D. and Nocedal, J.: On the limited memory BFGS method for large scale optimization, *Mathematical programming*, Vol. 45, No. 1, pp. 503–528 (1989).