局面情報からの探索信頼性の推定による将棋のProbCutの性能向上

亀甲 博貴[1]　**浦 晃**[2]　**三輪 誠**[3]
鶴岡 慶雅[2]　**近山 隆**[2]

[1] 東京大学工学部

[2] 東京大学大学院工学系研究科

[3] マンチェスター大学

2012.11.10

発表の流れ

1. 背景
2. 関連研究
3. 提案手法
4. 使用プログラムの説明
5. 評価
6. まとめ

発表の流れ

1. 背景
2. 関連研究
3. 提案手法
4. 使用プログラムの説明
5. 評価
6. まとめ

背景

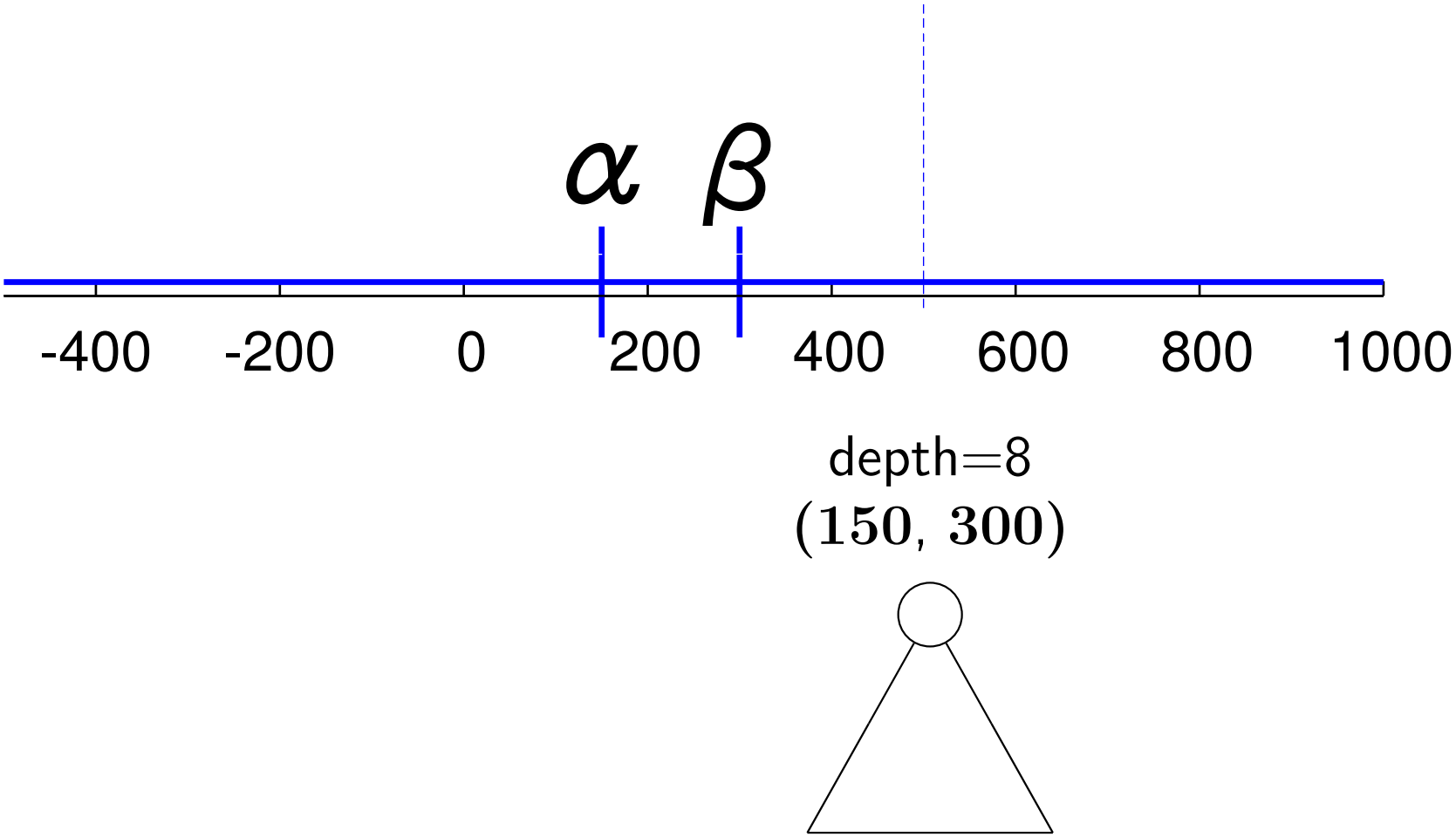
- 将棋の強いゲームプレイヤを作りたい
  - よい評価関数を持つ
  - 限られた時間で深く読める
  - etc...
- 結果に影響を与えない部分の探索を回避したい

  → 後ろ向き枝刈り
    - $\alpha\beta$ 探索 ...

  → 前向き枝刈り
    - ProbCut, Null Move Forward Pruning, Futility Pruning ...
- 本研究では ProbCut に着目する

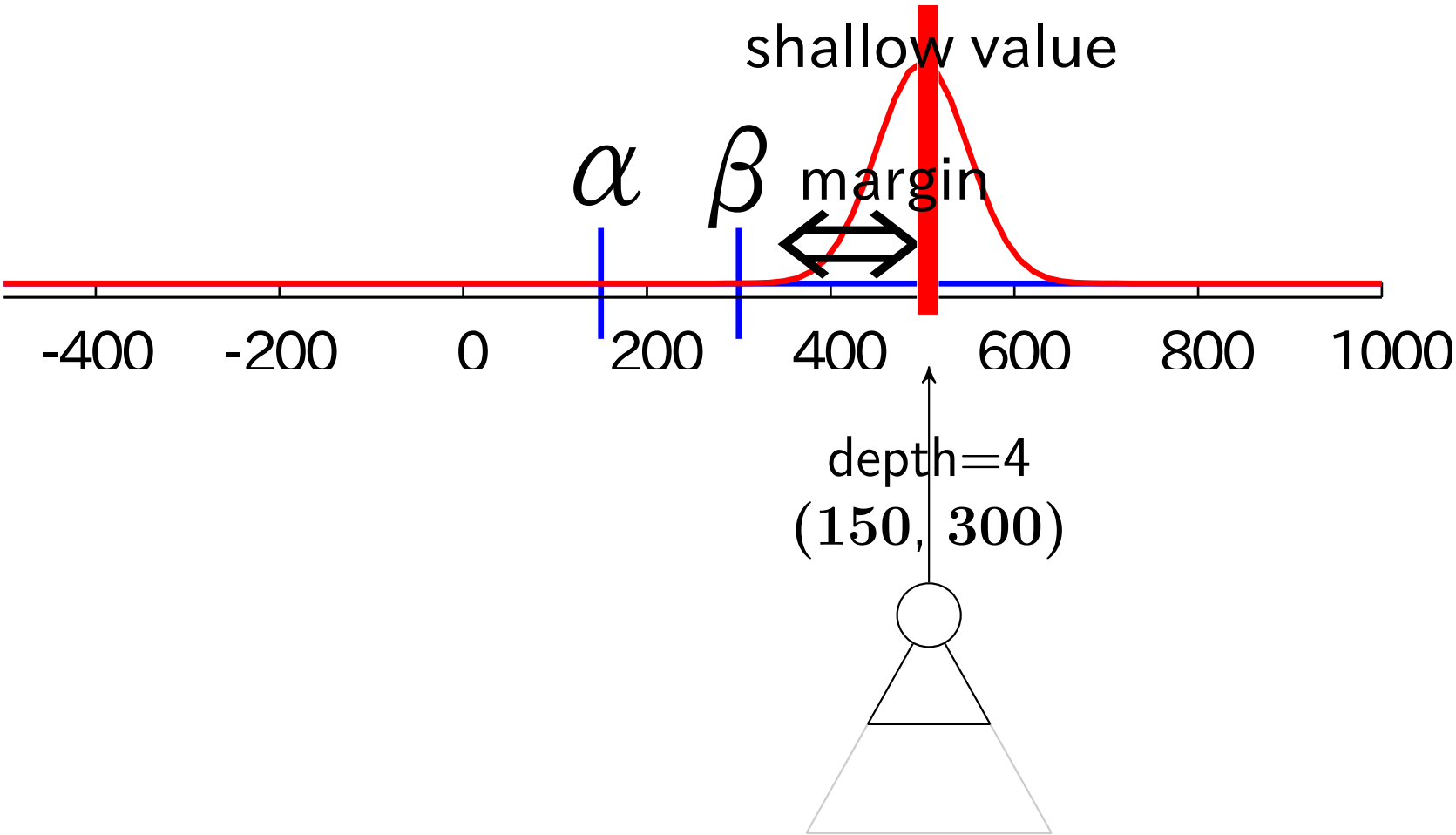
ProbCut[Buro, 1995]

浅い探索の結果と $\alpha\beta$Window を利用した枝刈り



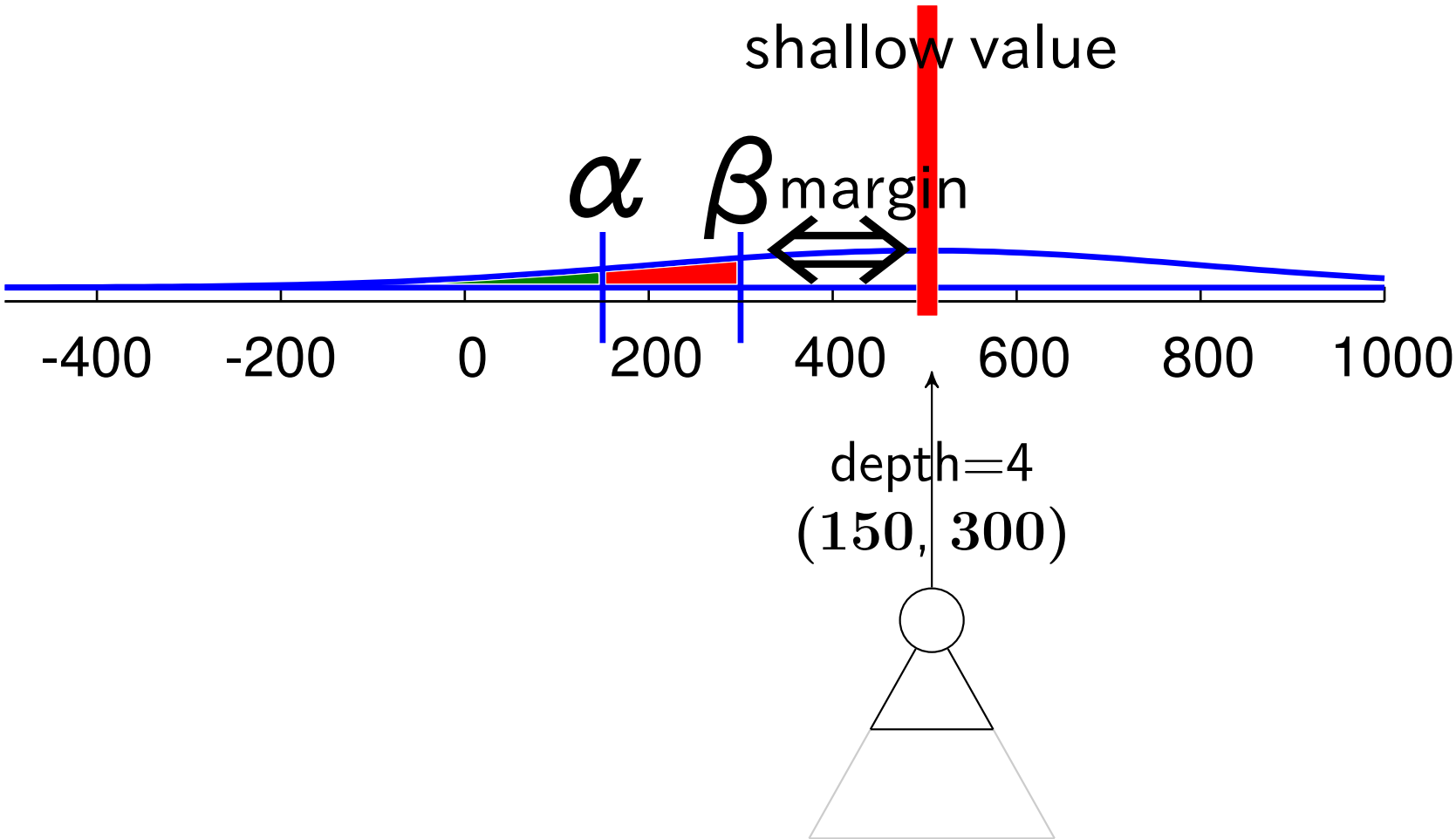
ProbCut[Buro, 1995]

浅い探索の結果と $\alpha\beta$Window を利用した枝刈り
分布がうまく予測できたら



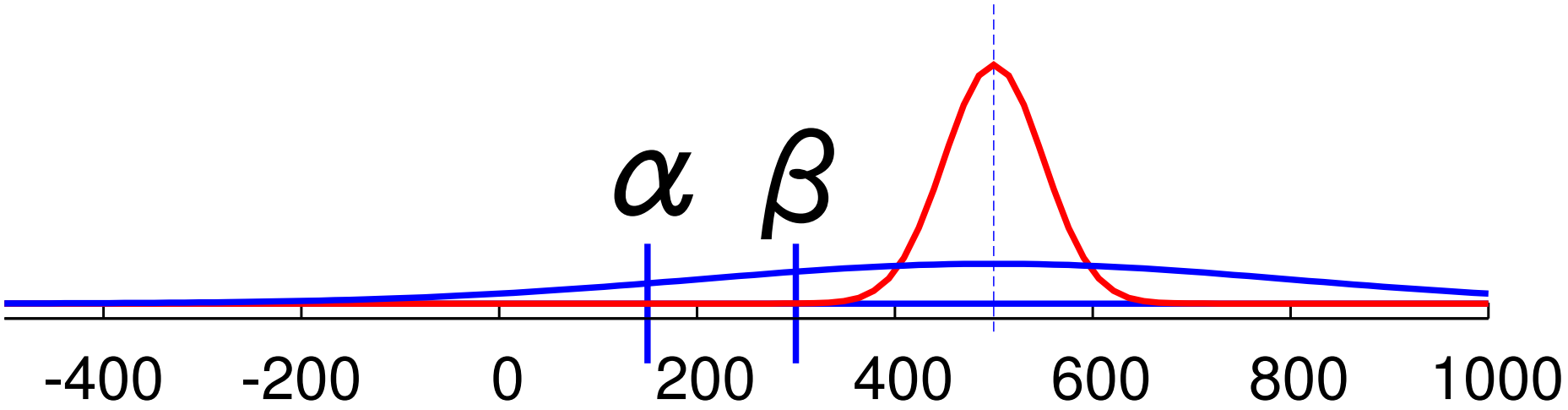
ProbCut[Buro, 1995]

浅い探索の結果と $\alpha\beta$Window を利用した枝刈り
分布が予測より広かったら



目的

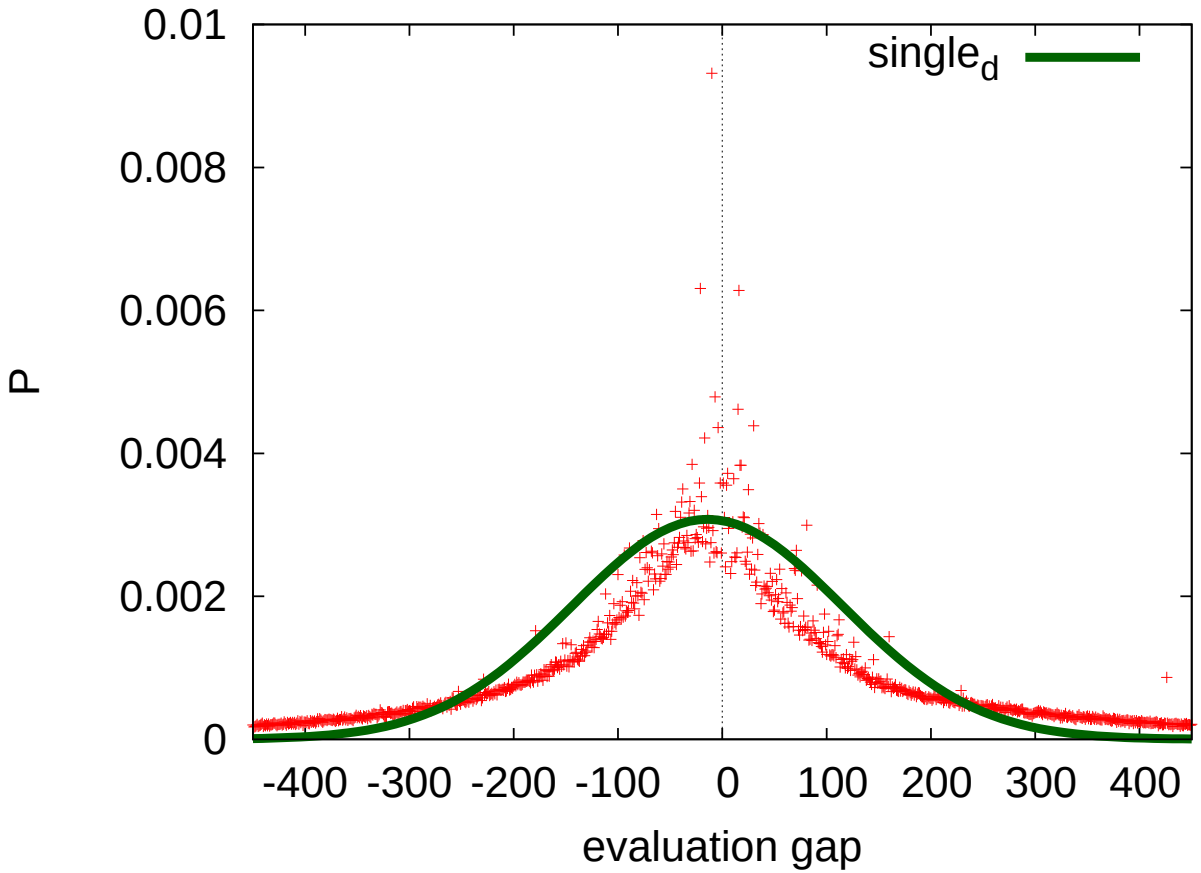
局面毎に分布を求めたい



深浅差 = 深い探索の評価値 − 浅い探索の評価値

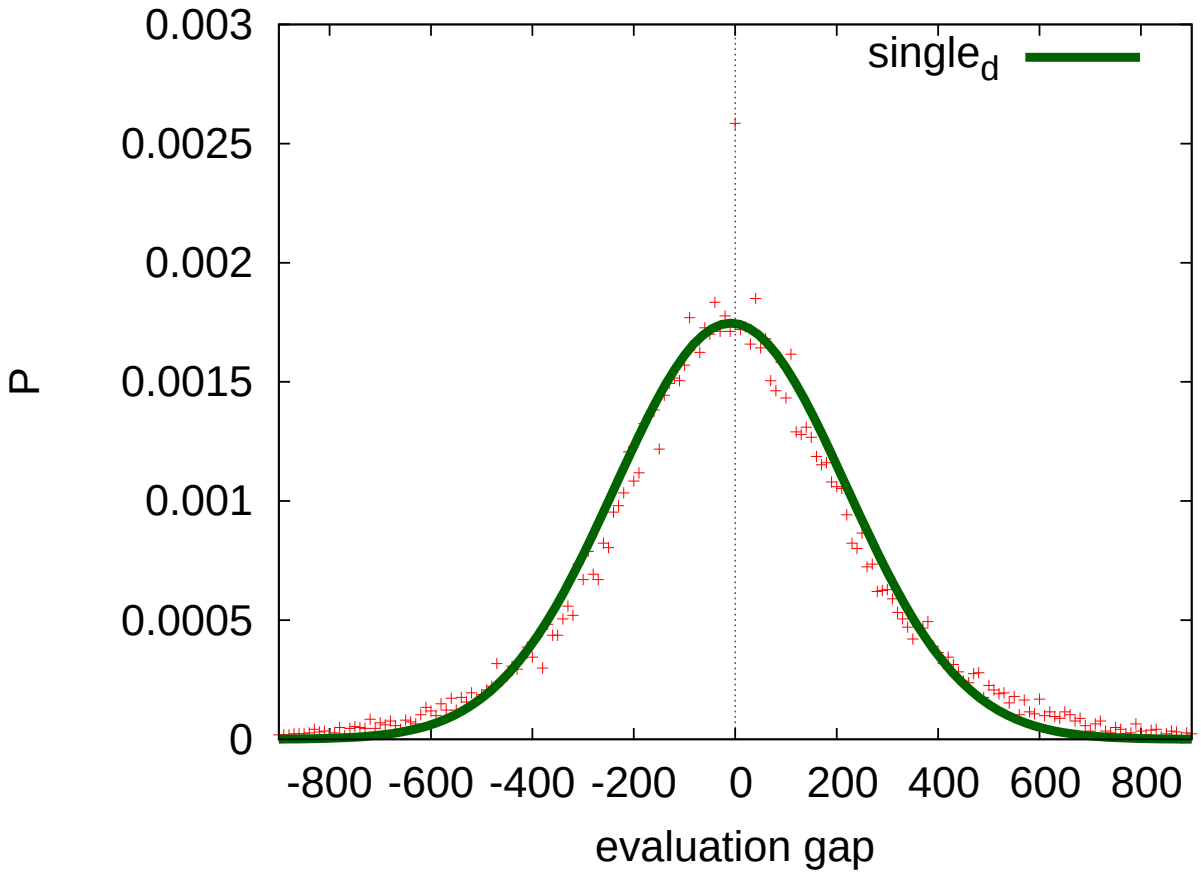
を定義し，その分布を推定する

深浅差の分布



- プロの棋譜に表れた局面での深浅差の分布
- 正規分布より**裾野が広い** →ProbCut には不都合

深浅差の分布



- 中盤の局面を抜き出しての分布
- 正規分布と近似できる

深浅差の分布

局面のある一部分を抜き出すと，深浅差の分布は正規分布に近似できる
期待値 $\mu = 0$ で，標準偏差 σ さえ与えれば一意に決まる

$\rightarrow$ 局面が属する正規分布を**局面毎に**与える

発表の流れ

1. 背景
2. **関連研究**
3. 提案手法
4. 使用プログラムの説明
5. 評価
6. まとめ

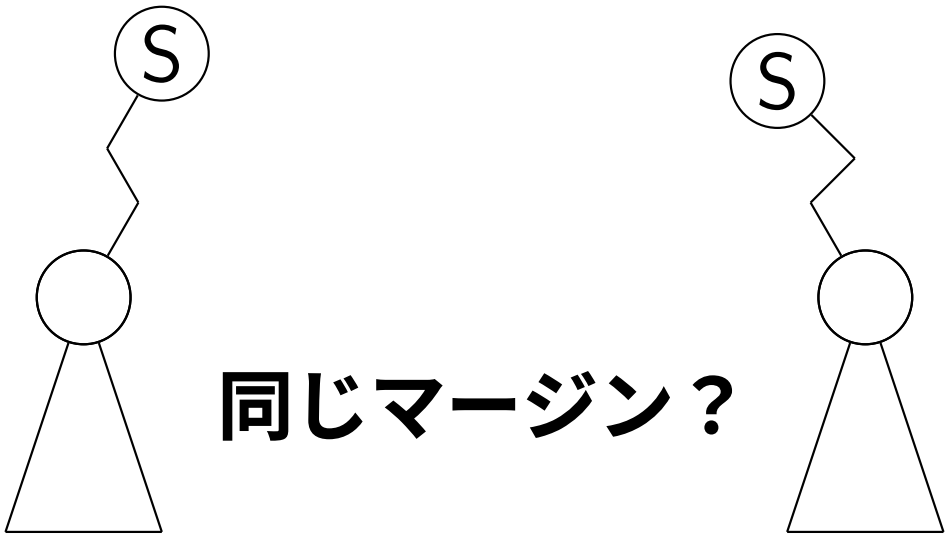
マージンを変動させる手法

- ProbCut の改良と将棋への適用 [吉原ら, 2002]
  - 初期局面からの手数でマージンを回帰
- 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 - [柴原ら, 2002]
  - ゲーム中のイベント毎に局面を分類

ゲームの進行（序盤・中盤・終盤などの概念）によってマージンを決定する

既存手法の問題点

- ゲームの進行の上で似たような位置にある局面では同じマージンを用いる
  - 中盤は中盤のマージンを用いた枝刈り
  - ゲームの上での性質が同じとは限らない



既存手法の問題点

- ゲームの進行の上で似たような位置にある局面では同じマージンを用いる
  - 中盤は中盤のマージンを用いた枝刈り
  - ゲームの上での性質が同じとは限らない
- 深浅差に強く影響する特徴がゲームの進行の他にもあるのではないか？
  - → **局面の特徴**を用いてよりよいマージンを決定したい

ゲームの進行を表す値として**進行度（0〜127）**を用いて，これと**局面の特徴**からマージンを決定する

発表の流れ

1. 背景
2. 関連研究
3. **提案手法**
4. 使用プログラムの説明
5. 評価
6. まとめ

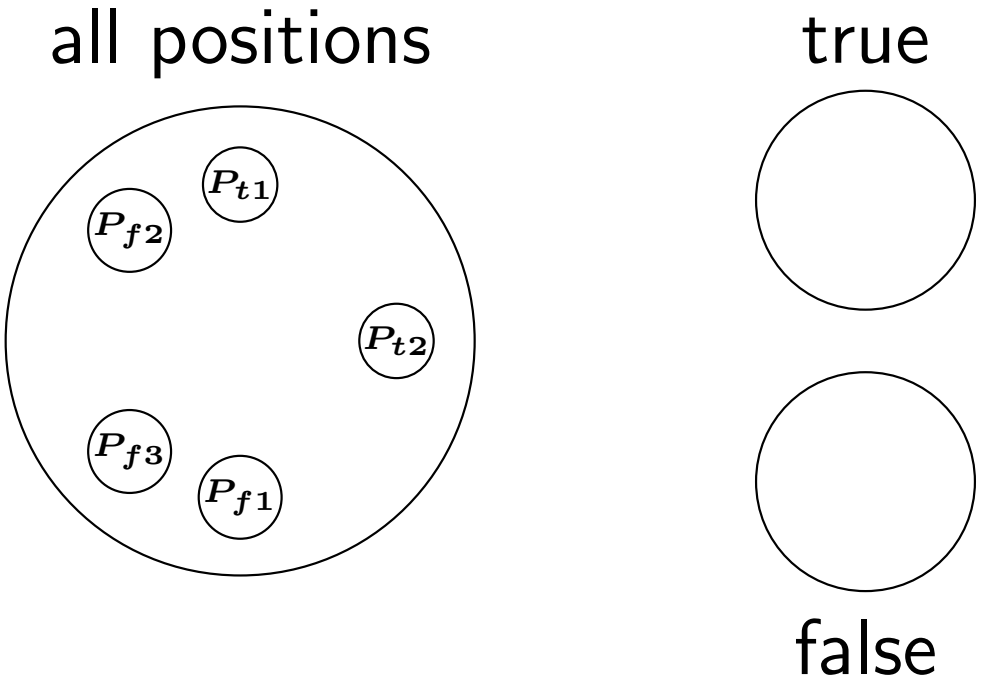
提案手法

- 特徴の決定
- 深浅差の分布の推定
  - **分類毎**に訓練データから標準偏差を計算して進行度との関係を決定する

    進行度として 0〜127 の整数値を取るものを用いる

特徴の決定

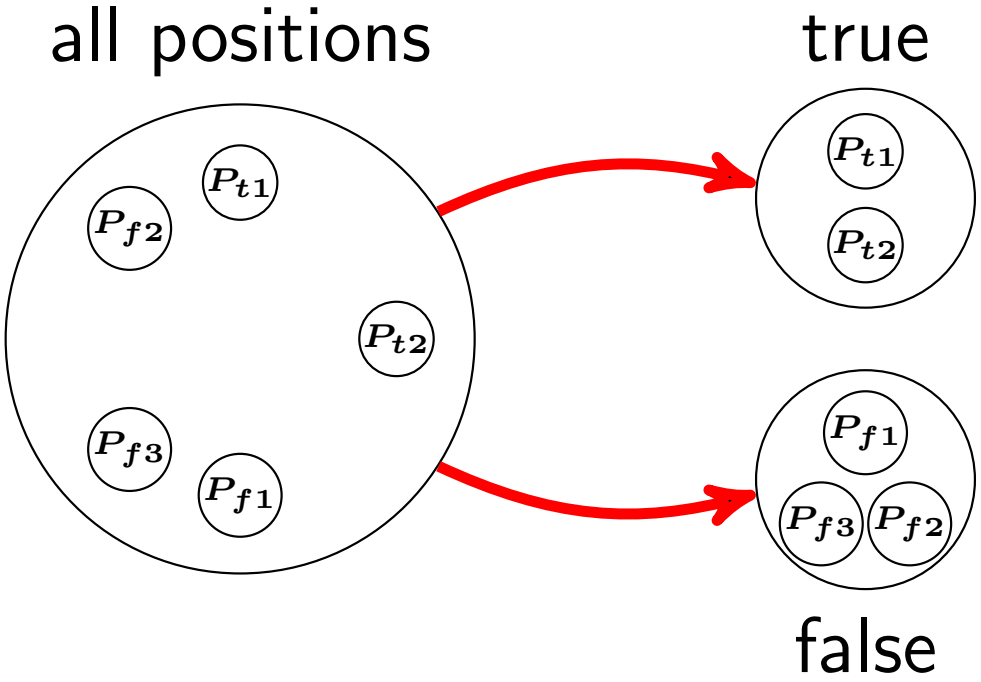
用いる特徴を決定する



- **評価関数が用いている特徴を候補とする**

特徴の決定

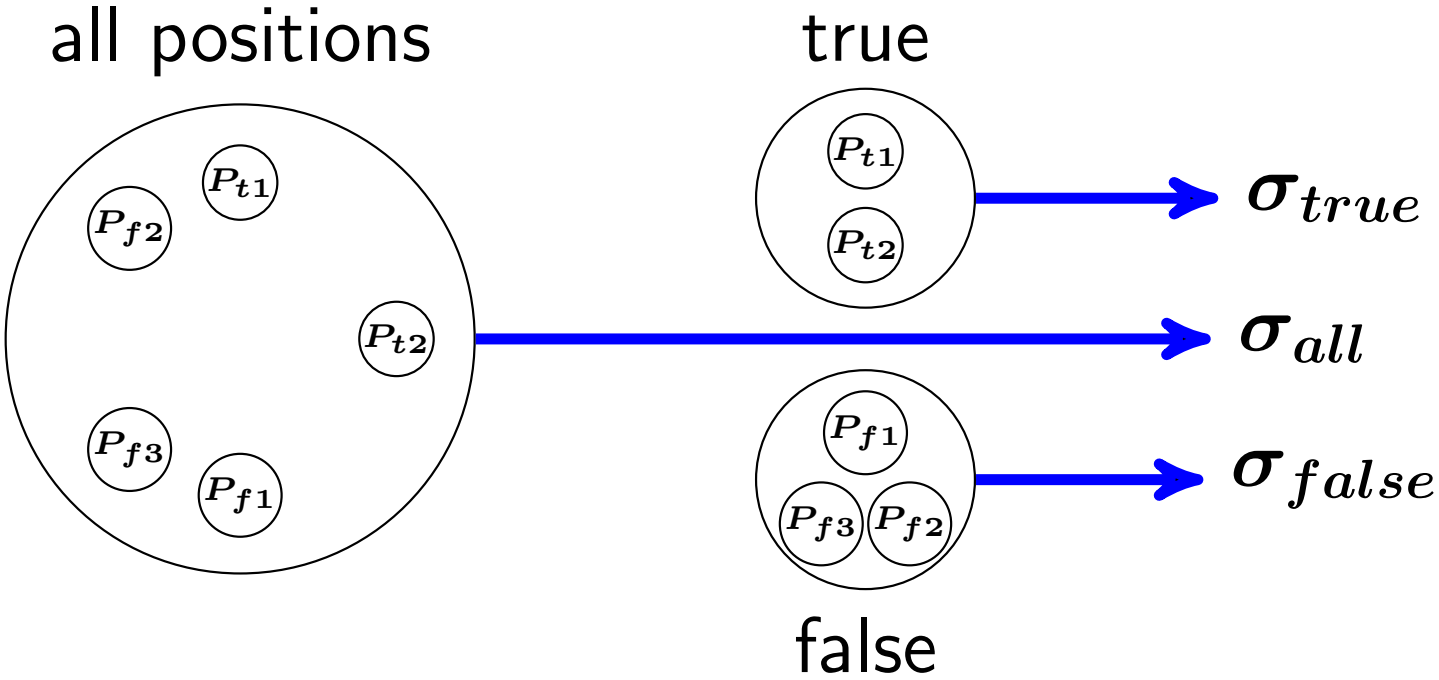
用いる特徴を決定する



- **特徴の有無で局面を 2 値分類**

特徴の決定

用いる特徴を決定する



- 2 つの局面群それぞれで標準偏差を計算
- それらが大きくずれていれば特徴を候補とする
- 一定の範囲内の進行度について行う

局面の分類

決定された N 個の特徴から N+1 クラスに分類する

A, $\neg A \wedge B$, $\neg A \wedge \neg B \wedge C$, ...

後述する 5 特徴を同時に持つ：1,288 / 802,881 局面
これを 128 分割して使用するのは心許ない……

- 標準偏差を計算するために分類毎に相当数のデータが必要
- 進行度毎に標準偏差を計算するため，実質 $(N+1) \times 128$ クラスの分類

発表の流れ

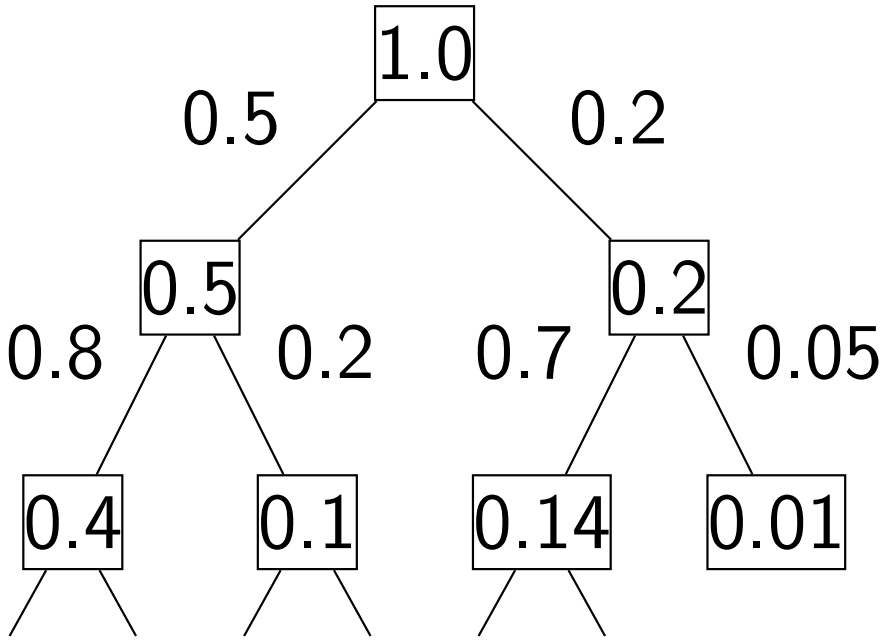
1. 背景
2. 関連研究
3. 提案手法
4. **使用プログラムの説明**
5. 評価
6. まとめ

使用プログラム「激指」

将棋プログラム「激指」を使用

- 実現確率探索や静止探索などの縮退・延長探索
- 各種の前向き枝刈りを行う
- 駒の位置から進行度を計算

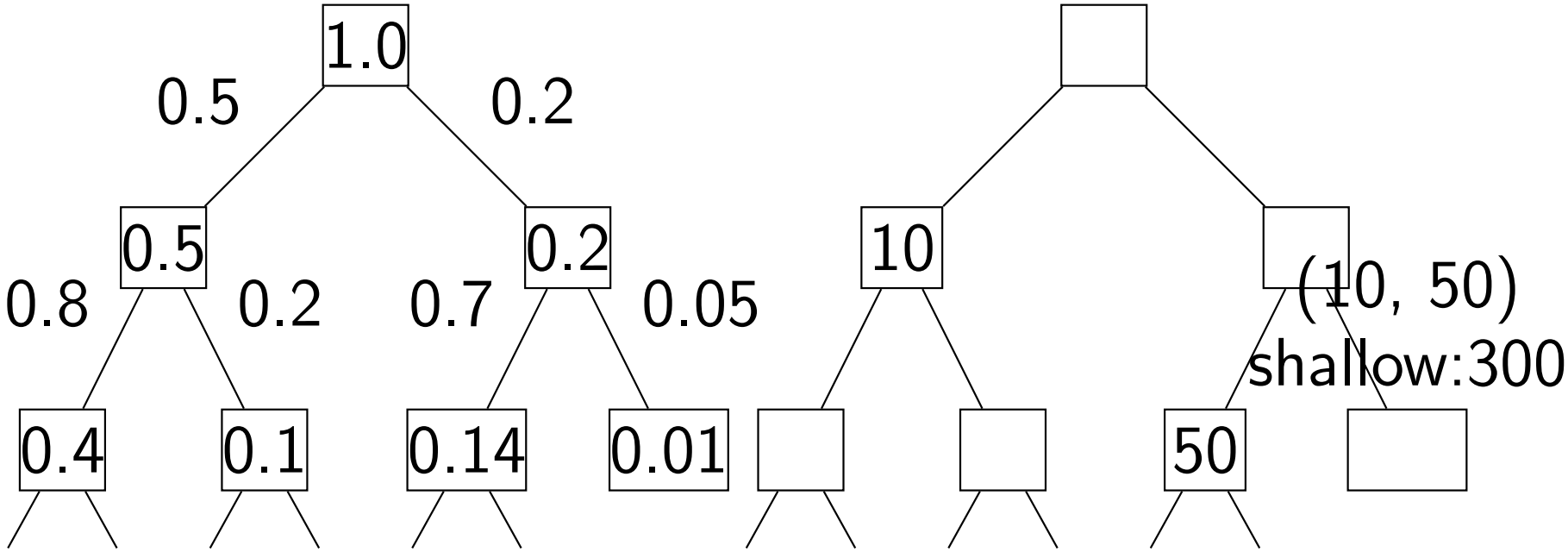
実現確率探索 [Tsuruoka et al., 2002]



実現確率に基づく探索

- **指し手毎に遷移確率を与え，掛けあわせたもの**
- **閾値を下回るまで探索を行う**

実現確率探索とProbCut



- **遷移確率は指し手毎に絶対評価として与えられる**
- ProbCut **は相対的に探索が不要である確率を基準に枝刈り**
  - **同時に実装して影響はないか**

その他の前向き枝刈り手法

激指は他の前向き枝刈り手法も用いている

- Null Move Forward Pruning

  パス手を選択した場合の探索

- Futility Pruning

  （激指では）リーフノード付近で探索を打ち切って評価してみる

ProbCut **を行うノードで** Null Move Forward Pruning **は同時に行われる**

進行度

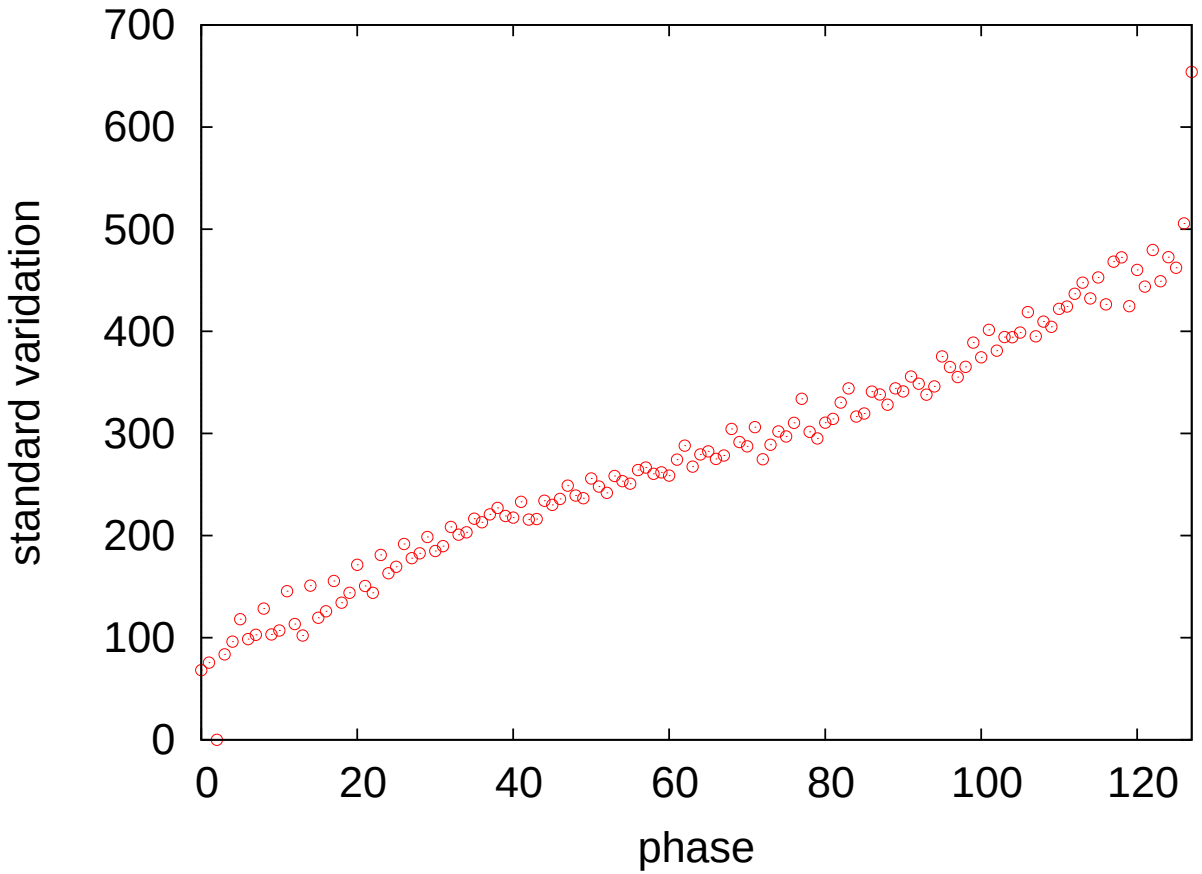
ゲームの進行具合を示す

- 0-127 の整数値
- 初期局面は 0 で，大きいほど終局に近い
- { 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
  持ち駒であるか } で計算

左図の赤い駒が進行度に影響する

進行度 45

標準偏差の進行度による近似



進行度で標準偏差を直線に近似できるようにみえる

発表の流れ

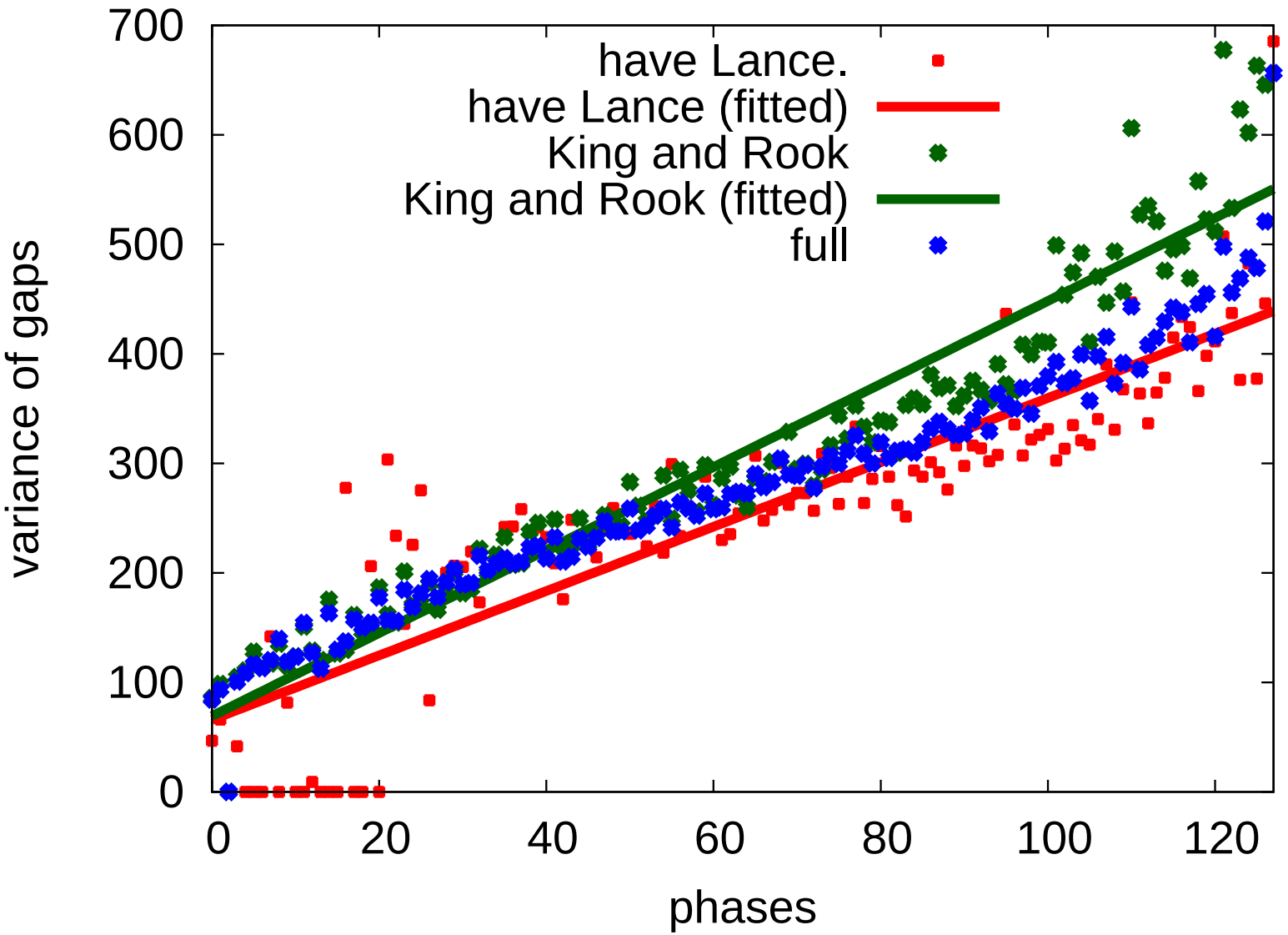
1. 背景
2. 関連研究
3. 提案手法
4. 使用プログラムの説明
5. **評価**
6. まとめ

用いる特徴

特徴として以下の 5 つを用いる

- 持ち駒に香車がある
- 飛車と自玉が近い
- 角が敵の金銀に効いている
- 角が相手玉に近い
- 角が 5 五から離れたところにある

局面の分類


have Lance.
have Lance (fitted)
King and Rook
King and Rook (fitted)
full
variance of gaps
phases
0
100
200
300
400
500
600
700
0
20
40
60
80
100
120

ProbCut に導入しての評価

以下では激指を用いて評価を行った

- ProbCut を実装し，マージンを変動させた
- 比較対象として
  - 提案手法（変動 ProbCut）
  - ProbCut を行わない
  - 固定マージンで ProbCut（固定 ProbCut）
  - 進行度のみからマージンを変動させて ProbCut（進行度 ProbCut）

予稿の段階から式を更新しました

を挙げた

探索ノード数の比較

- プロの棋譜から抜き出した 1,000 局面を探索
- 深さ 13 で固定
- プロの指し手との一致率と，探索に要したノード数・時間を比較した

探索ノード数の比較（固定マージン）

固定マージンの場合

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
固定 (100)	552	2,049	147M（36.6）	71,795
固定 (300)	564	2,646	205M（51.0）	77,514
固定 (800)	569	3,272	303M（75.4）	92,670
固定 (1500)	570	4,107	386M（96.0）	94,205
固定 (2000)	562	4,347	417M（104.7）	96,115

ノード数は大きく削減できている

探索ノード数の比較（固定マージン）

固定マージンの場合

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
固定 (100)	552	2,049	147M（36.6）	71,795
固定 (300)	564	2,646	205M（51.0）	77,514
固定 (800)	569	3,272	303M（75.4）	92,670
固定 (1500)	570	4,107	386M（96.0）	94,205
固定 (2000)	562	4,347	417M（104.7）	96,115

一致数が減少している

探索ノード数の比較（変動マージン）

変動マージンの場合（σ は求めた標準偏差）

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
変動 (1.0σ)	558	2,640	209M（52.0）	78,222
変動 (2.0σ)	567	3,315	303M（75.4）	91,495
変動 (3.0σ)	591	3,701	362M（90.0）	97,725
変動 (4.0σ)	578	3,790	389M（96.8）	102,691

ノード数は削減できている

探索ノード数の比較（変動マージン）

変動マージンの場合（σ は求めた標準偏差）

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
変動 (1.0σ)	558	2,640	209M（52.0）	78,222
変動 (2.0σ)	567	3,315	303M（75.4）	91,495
変動 (3.0σ)	591	3,701	362M（90.0）	97,725
変動 (4.0σ)	578	3,790	389M（96.8）	102,691

3.0σ では一致数の上昇が見られる

探索ノード数の比較（進行度のみ）

進行度のみを用いた変動マージンの場合

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
進行度 (1.0σ)	550	2,731	227M（56.5）	83,021
進行度 (2.0σ)	567	3,247	319M（79.4）	98,223
進行度 (3.0σ)	575	3,801	382M（95.0）	100,437

ノード数は削減できている

探索ノード数の比較（進行度のみ）

進行度のみを用いた変動マージンの場合

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
進行度 (1.0σ)	550	2,731	227M（56.5）	83,021
進行度 (2.0σ)	567	3,247	319M（79.4）	98,223
進行度 (3.0σ)	575	3,801	382M（95.0）	100,437

3.0σ では一致数が ProbCut 無しに近い値を示した

探索ノード数の比較（手法間の比較）

探索ノード数75%カット付近の比較

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
固定 (800)	569	3,272	303M（75.4）	92,670
変動 (2.0σ)	567	3,315	303M（75.4）	91,495
進行度 (2.0σ)	567	3,247	319M（79.4）	98,223

手法間で性能差があまり見られない

探索ノード数の比較（手法間の比較）

一致数ベストの比較

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
固定 (1500)	570	4,107	386M（96.0）	94,205
変動 (3.0σ)	591	3,701	362M（90.0）	97,725
進行度 (3.0σ)	575	3,801	382M（95.0）	100,437

変動マージンの結果が探索ノード数の削減量に対し，よい結果を得ている

探索ノード数の比較（変動マージン）

変動マージンの場合（σ は求めた標準偏差）

	一致数	時間（s）	ノード数（%）	#/s
ProbCut なし	574	4,294	402M（100）	93,722
変動 (1.0σ)	558	2,640	209M（52.0）	78,222
変動 (2.0σ)	567	3,315	303M（75.4）	91,495
変動 (3.0σ)	591	3,701	362M（90.0）	97,725
変動 (4.0σ)	578	3,790	389M（96.8）	102,691

マージンを大きくする = 枝刈りの条件を厳しくすると時間当たりの探索ノード数が増える

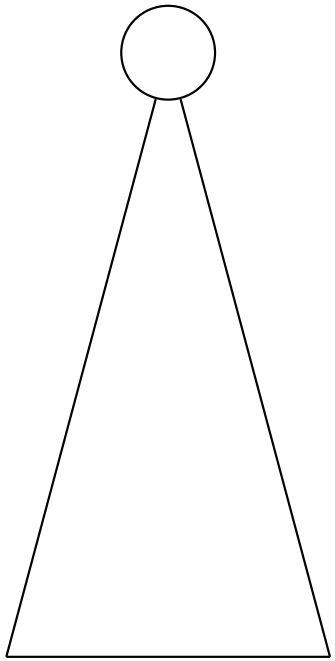
時間当たりの探索ノード数の考察

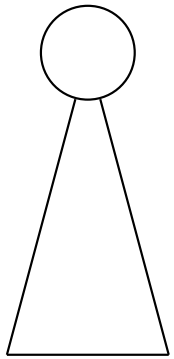
- 実行時間の約 25%は評価関数
  - マージンの決定関数はほぼ無視できる

  → 探索ノード数当たりのリーフノード数が多いと減少

ProbCut の影響

- ProbCut によりカットが発生
  - 探索ノード数 $O(n^8) \rightarrow O(n^4)$ に減少
    - $\rightarrow$ 探索ノード数当たりのリーフノード数が多い

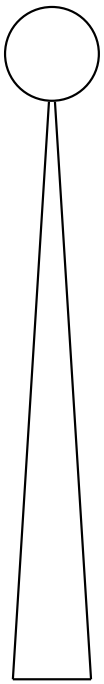




Null Window 探索の影響

- Null Window 探索
  - 探索木は縦長

  → 探索ノード数当たりのリーフノード数が少ない
  呼べば呼ぶほど探索ノード数当たりのリーフノード数が少なくなる



自己対戦による棋力の比較

- プロの棋譜を 35 手読み，先後を入れ替えて 2 対戦
- 210 局面・420 対戦を行い勝率を比較した
- 1 手 3 秒で固定
- 200 手（お互いに 100 手づつ）指して決着が付かなければ引き分けとし，0.5 勝 0.5 敗扱い
- 224.5 勝（53.4%）以上で 5%有意

変動マージン vsProbCut 無し

プレイヤ	結果（win-draw-lose）
変動 1.0σ vs ProbCut 無し	167-32-221 （0.436）
変動 2.0σ vs ProbCut 無し	185-17-218 （0.460）
変動 3.0σ vs ProbCut 無し	209-23-188 （0.525）
変動 4.0σ vs ProbCut 無し	192-23-205 （0.485）

適切なマージンで ProbCut 無しに勝ち越した

- **しかし有意ではなく，棋力差があるとは言えない**

固定マージンvsProbCut無し

プレイヤ	結果（win-draw-lose）
固定300 vs ProbCut無し	141-32-243 （0.374）
固定800 vs ProbCut無し	209-27-184 （0.530）

有意に勝ち越しているとは言えない

変動マージンvs固定マージン

プレイヤ	結果（win-draw-lose）
変動 3.0σ vs 固定 800	205-20-195 （0.512）

この設定では棋力差が見られない

発表の流れ

1. 背景
2. 関連研究
3. 提案手法
4. 使用プログラムの説明
5. 評価
6. まとめ

まとめ

- 局面の特徴が深浅差の分布に影響を与えている
- 他の枝刈り手法と同時に実装しても ProbCut は有効かもしれない
- ProbCut への適用によって有意な棋力差は見られない

  有意でない範囲で勝ち越しており，設定を変えれば有意な棋力差が見られる可能性もある

今後の課題

訓練データから標準偏差を直接計算する手法の有効性が見られない

→ 1 局面毎に標準偏差を与えるようなシステムがあるとよい

最尤推定によって得られないか