情報論理工学
研究室

第9回：
種々の探索

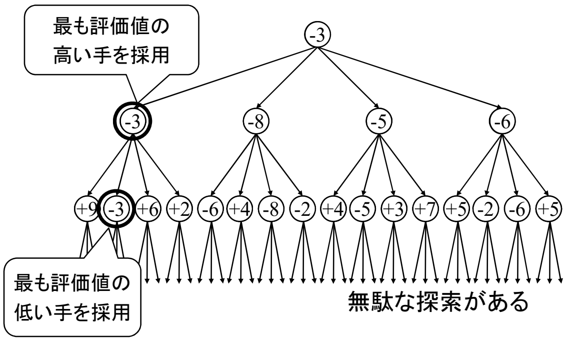
ミニマックス(mini-max)法

- ミニマックス法
  - 自分にとっての最善手＝相手にとっての最悪手
    (二人零和ゲームの場合)

  ⇒ 相手が常に最善手を指してくると仮定

自分の手番：最も評価値の高い手を採用
相手の手番：最も評価値の低い手を採用

ミニマックス法



無駄な探索がある

アルファベータ(alpha-beta)法

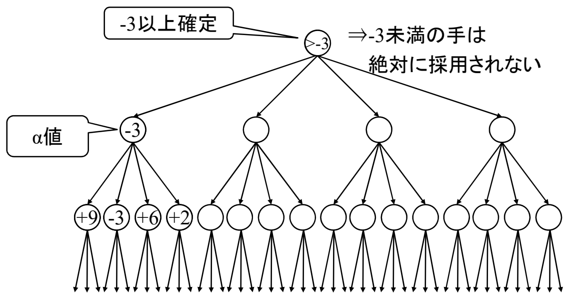
- アルファベータ法
  - ミニマックス法の改良アルゴリズム
    - 必要の無い探索は行わない
    - 絶対に採用されない手は読まない

α：それまでに発見した自番で最も大きな評価値
β：それまでに発見した相手番で最も小さい評価値

相手の手番：αよりも小さい評価値になれば探索打ち切り
自分の手番：βよりも大きい評価値になれば探索打ち切り

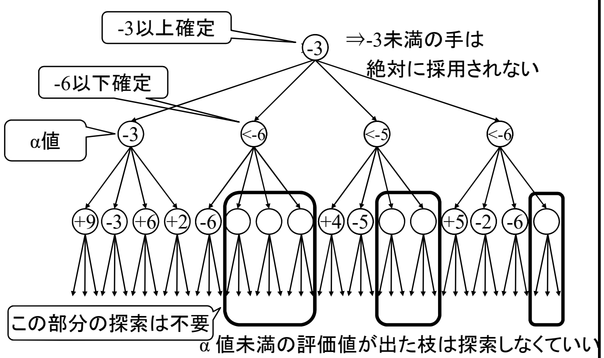
α以上β以下の手を探索する

アルファベータ法

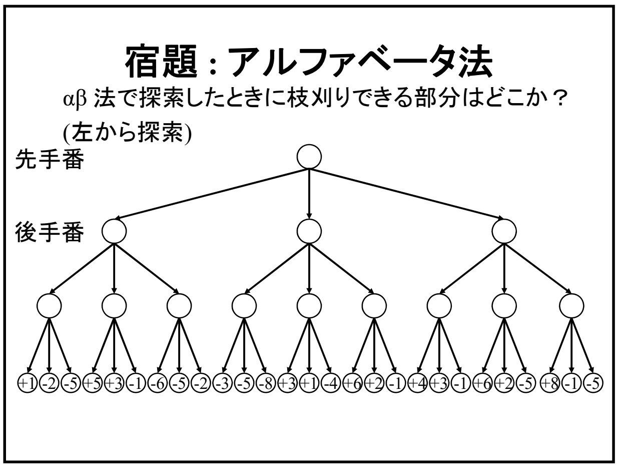


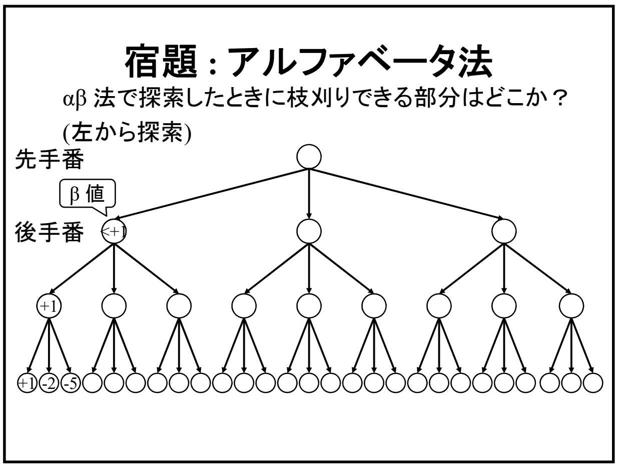
α値未満の頂点が現れると探索を打ち切る

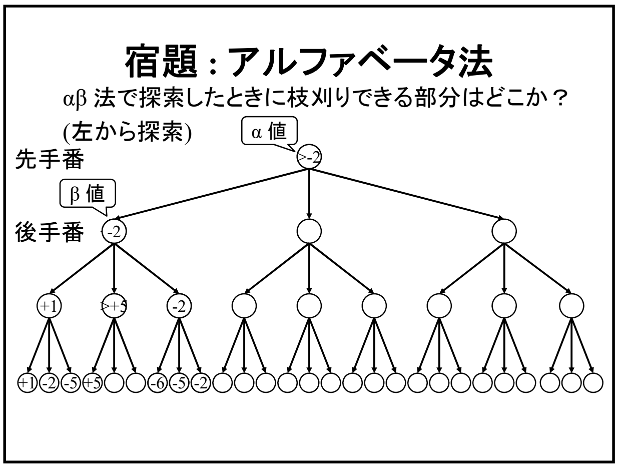
アルファベータ法

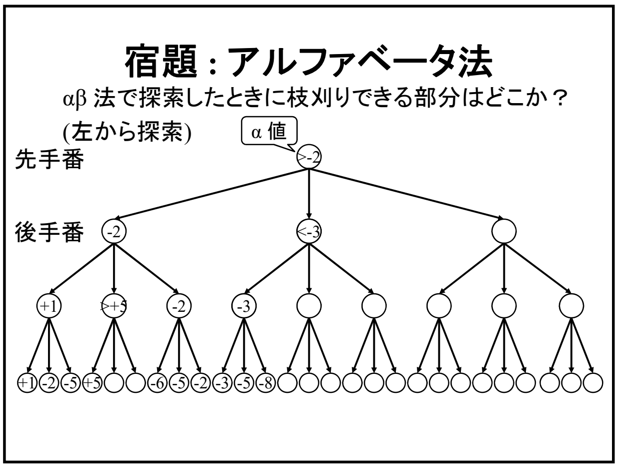


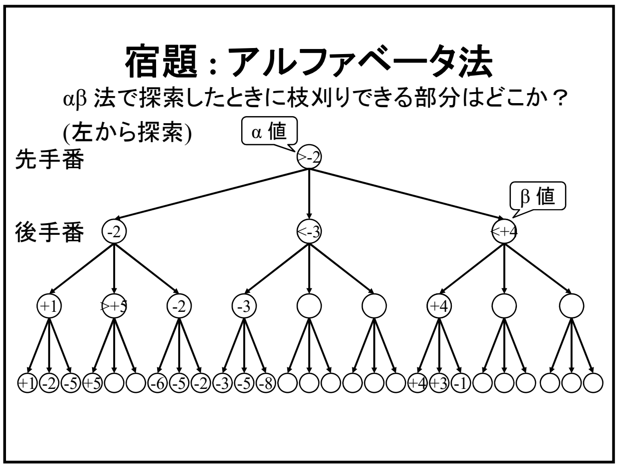
α値未満の評価値が出た枝は探索しなくていい

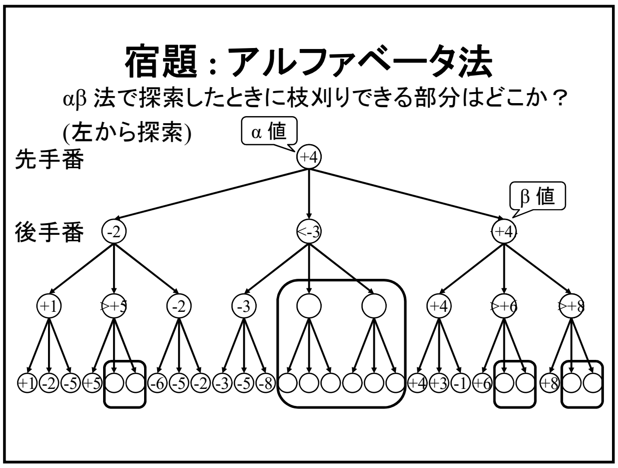

宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
先手番
後手番
+1 -2 -5 +5 +3 -1 -6 -5 -2 -3 -5 -8 +3 +1 -4 +6 +2 -1 +4 +3 -1 +6 +2 -5 +8 -1 -5


宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
先手番
β 値
後手番
<+1
+1
+1 -2 -5


宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
α 値
先手番
>-2
β 値
後手番
-2
+1 >+5 -2
+1 -2 -5 +5 -6 -5 -2


宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
α 値
先手番
>-2
後手番
-2 <-3
+1 >+5 -2 -3
+1 -2 -5 +5 -6 -5 -2 -3 -5 -8


宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
α 値
先手番
>-2
β 値
後手番
-2 <-3 <+4
+1 >+5 -2 -3 +4
+1 -2 -5 +5 -6 -5 -2 -3 -5 -8 +4 +3 -1


宿題 : アルファベータ法
αβ 法で探索したときに枝刈りできる部分はどこか？
(左から探索)
α 値
先手番
+4
β 値
後手番
-2 <-3 +4
+1 >+5 -2 -3 +4 >+6 >+8
+1 -2 -5 +5 -6 -5 -2 -3 -5 -8 +4 +3 -1 +6 +8

アルファベータ法の計算量

b : 各頂点での分岐数
d : 探索の深さ

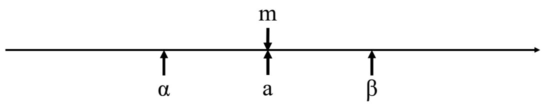
ミニマックス法 :　b^{d}
アルファベータ法 : $b^{d/2}$ (最も条件が良い場合)

同じ時間で2倍の深さまで読める

アルファベータ法の返り値

m : ミニマックス法の返り値
a : アルファベータ法の返り値

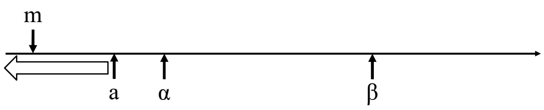
$\alpha < m < \beta$ のとき $a = m$



アルファベータ法の返り値

m : ミニマックス法の返り値
a : アルファベータ法の返り値

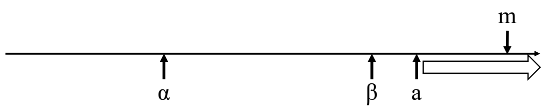
$m \leqq \alpha$ のとき　$m \leqq a \leqq \alpha$



アルファベータ法の返り値

m : ミニマックス法の返り値
a : アルファベータ法の返り値

$\beta \leqq m$ のとき　$\beta \leqq a \leqq m$



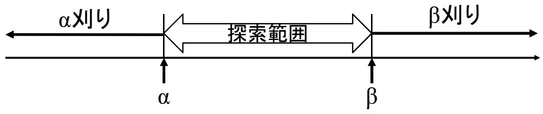
アルファベータ法の返り値

m : ミニマックス法の返り値
a : アルファベータ法の返り値

		返り値から分かること
$m \leqq \alpha$ のとき	$m \leqq a \leqq \alpha$	α 値以下
$\alpha < m < \beta$ のとき	$a = m$	正しい値
$\beta \leqq m$ のとき	$\beta \leqq a \leqq m$	β 値以上

アルファベータ法の特徴

- アルファベータ法の特徴
  - $\alpha < m < \beta$ の範囲に絞って探索
  - 範囲外では枝刈り

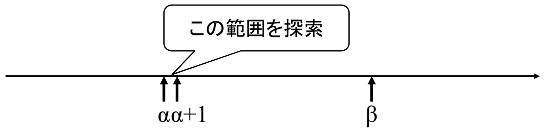


|β-α|が狭いと枝狩りされやすい

Scout法

- Scout法
  - 狭い範囲で探索して評価値の範囲を見積もる

β'=α+1 として探索



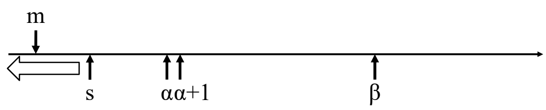
評価値がα値より小さいか否かを高速に判定できる

(※) 範囲(α, α+1)に整数は無い

Scout法の返り値

m: ミニマックス法の返り値
s: Scout法の返り値

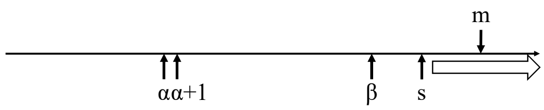
s≦αのとき　　m≦s≦α　　⇒α刈り



Scout法の返り値

m: ミニマックス法の返り値
s: Scout法の返り値

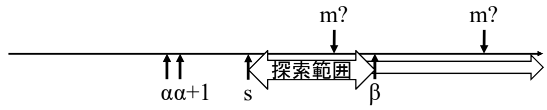
β≦nのとき　　β≦s≦m　　⇒β刈り



Scout法の返り値

m: ミニマックス法の返り値
s: Scout法の返り値

α<n<βのとき　　α<s≦m



範囲(s,β)で再探索

(α,β)より狭いので高速(かもしれない)

```
/* スカウト法により局面の評価値を計算する */
int scout (int depth, int alpha, int beta) {
  if (depth == 0) // 深さ制限に達した場合
    return value; // 先読み無しの評価値を返す
  int a = alpha, b = beta, s;
  ArrayList<Move> moveList = generateMoves(); // 合法手リスト生成
  for (Move move : moveList ) { // 全ての合法手に対して判定
    Phase phase = nextPhase (move); // 次の局面を生成
    s = -scout (depth-1, -b, -a);
    if (a<s && s<beta && depth <=2) { // α<s<βの場合
      a = -scout (depth-1, -beta, -s); // 範囲(s, β)で再探索
    }
    if (s>a) a=s; // α値更新
    if (a>=beta) return a; // β刈り
    b = a+1; // β値更新
  }
  return a;
}
```

同一局面の処理

- 探索中同一局面が現れるケース
  - 手順前後の同一局面
    - 手順が違っても同一局面となる
      ⇒局面の評価値を再利用できる
  - 千日手
    - 手順中で同一局面が現れる(千日手)
      ⇒探索の無限ループを回避する必要がある

局面の同一判定

- equals()メソッド
  - 同一の局面か判定する

```
boolean equals (Phase phase) {
  for (int i=0; i<SIZE; ++i)
    for (int j=0; j<SIZE; ++j)
      if (this.board[i][j] ≠ phase.board[i][j])
        return false; // 1箇所でも異なればfalse
  if (this.turn ≠phase.turn) return false;
                  :
  return true; // 全て同じならtrue
}
```

だがこの判定は時間がかかる

局面の同一判定

- 探索中には多くの局面が現れる
- 局面の同一判定は時間がかかる

⇩

同一の可能性のある局面を絞り込む

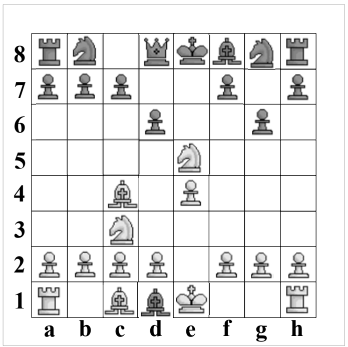
ハッシュ関数による同一判定
ハッシュ関数で局面を数値化、
同一のハッシュ値を持つ局面のみ同一判定

ハッシュ関数の例：チェス

駒がある：1
駒が無い：0
として64ビットの
数値で表現

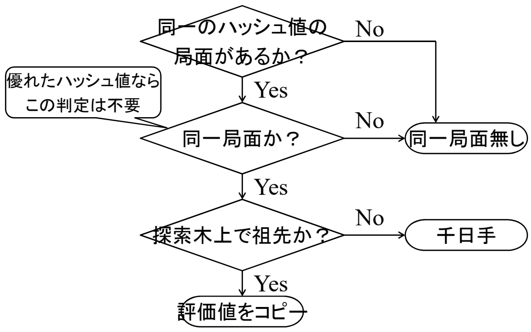
11011111
11100101
00010010
00001000
00101000
0010000
11111111
10111001

DFE51208
2820F7B9



(※)実際はもっと良いハッシュ関数を用いる

同一局面の判定



ハッシュ関数の作成

- ハッシュ関数の作成
  - 各マスに対して、駒の種類数分の乱数値を割当
  - 全てのマスの排他的論理和を取る

例:3目並べ

7	8	9
4	5	6
1	2	3

○番	000000
×番	111111

乱数値

	1	2	3	4	5	6	7	8	9
○	110101	011110	011010	010101	010011	101001	001100	000111	110110
×	011111	000010	101010	101000	100101	011000	011010	001010	100010
空	000000	000000	000000	000000	000000	000000	000000	000000	00000

局面のハッシュ関数

○番	000000
×番	111111

	1	2	3	4	5	6	7	8	9
○	110101	011110	011010	010101	$\boxed{010011}$	101001	001100	000111	$\boxed{110110}$
×	$\boxed{011111}$	000010	101010	101000	100101	011000	011010	001010	100010
空	000000	000000	000000	000000	000000	000000	000000	000000	00000

○番

	○	
×		

×番

		○
	○	
×		

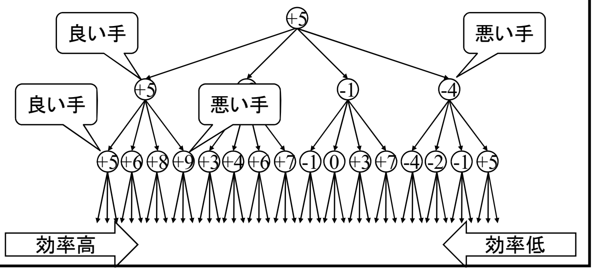
011111⊕010011⊕00000
=001100

011111⊕010011⊕110110⊕111111
=000101

ビット数が十分大きければハッシュ値の衝突は起こらない

アルファベータ法の効率

- アルファベータ法の効率
  - 良い手から先に探索すると効率がいい



アルファベータ法の効率

- アルファベータ法の効率
  - 良い手から先に探索すると効率がいい

でもどうやって「良い手」を探す？

そもそも「良い手」がわかるなら探索の必要無し

「良さそうな手」から先に探索

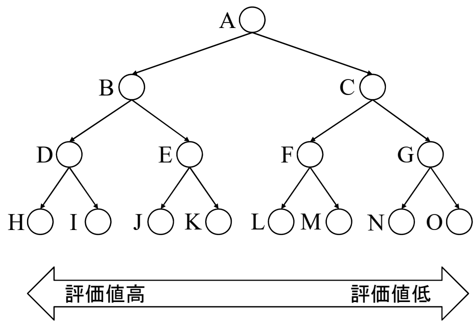
反復深化法

- 反復深化法
  - 探索の範囲を徐々に深くしていく

毎回評価値の高い順に枝を並べ替える
⇒アルファベータ法を効率良く行える

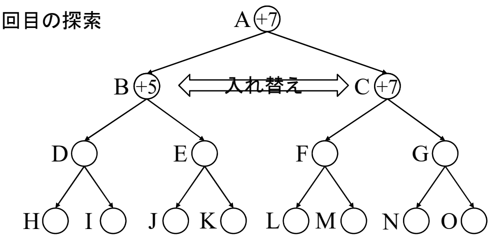
探索回数は増えるが、枝刈りできる利点が大きい

反復深化法



反復深化法

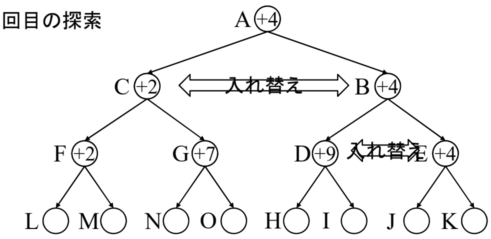
1回目の探索



Cの方が評価値が高い⇒BとCを入れ替える

反復深化法

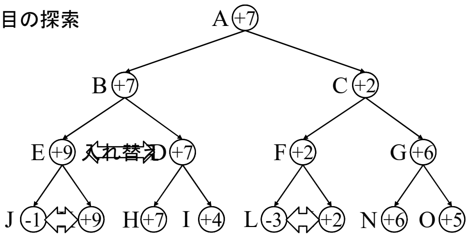
2回目の探索



Bの方が評価値が高い⇒BとCを入れ替える
Eの方が評価値が低い⇒DとEを入れ替える

反復深化法

3回目の探索



反復深化法

- 反復深化法
  - 浅い局面から順に探索
  - 探索した局面の情報は記憶しておく
    - 局面のハッシュ値
    - 最善手
    - 評価値
    - その評価値が真の値か、α値β値か
    - その評価値を得たときの探索の深さ

:

水平線効果

- 水平線効果
  - 探索で読む深さは有限

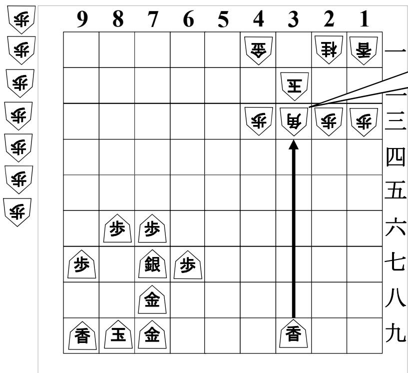
  ⇒より先に不利な局面があってもわからない

例:5手先まで読む場合

× 手A : 4手先で自玉が詰む
? 手B : 8手先で自玉が詰む
? 手C : 自玉は当面詰まない

手Bと手Cの
どちらがいいか
分からない

水平線効果:将棋



▲3九香まで

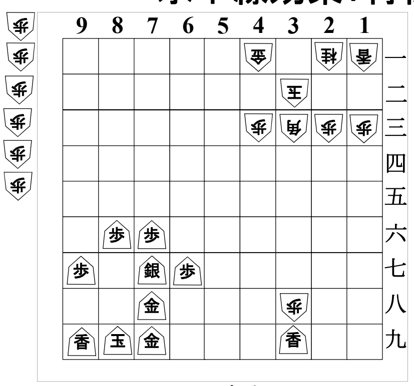
角は逃げられない！

この状態では
▲3三香成は
防げない

⇒角は諦めて
他の手を考えるべき

しかし…

水平線効果:将棋

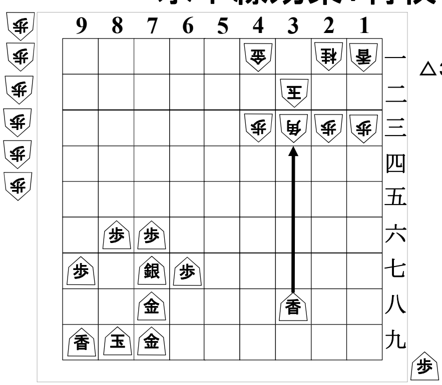


△3八歩まで

△3八歩

無意味な歩打ち

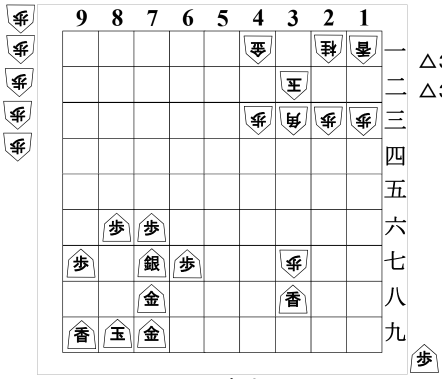
水平線効果:将棋



▲3八同香まで

△3八歩 ▲同香

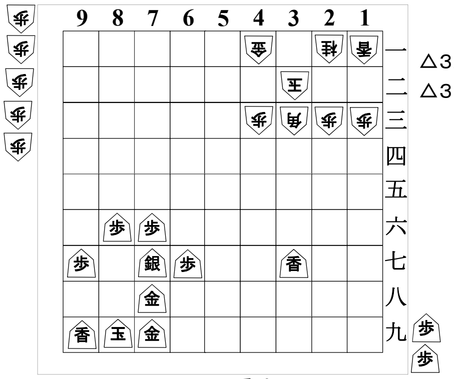
水平線効果：将棋



△3八歩　▲同香
△3七歩

△3七歩まで

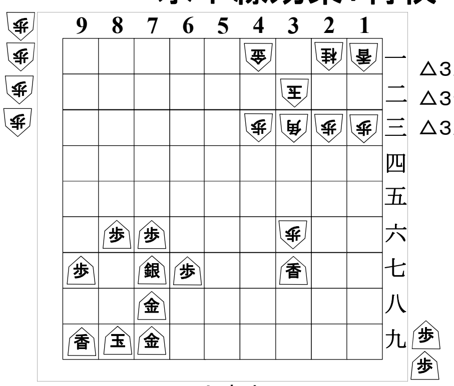
水平線効果：将棋



△3八歩　▲同香
△3七歩　▲同香

▲3七同香まで

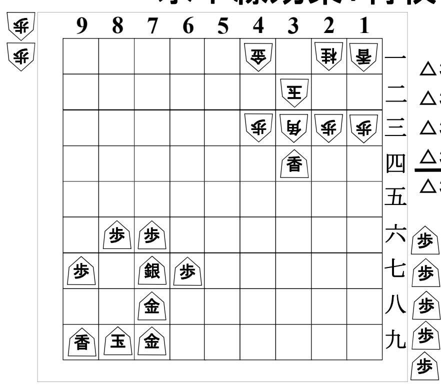
水平線効果：将棋



△3八歩　▲同香
△3七歩　▲同香
△3六歩

△3六歩まで

水平線効果：将棋

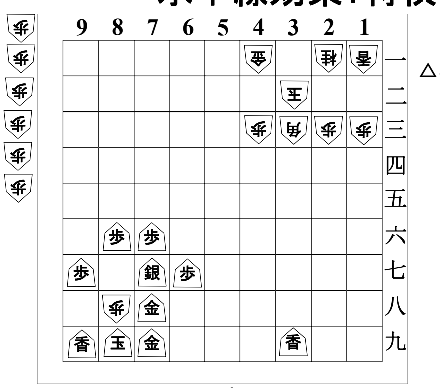


△3八歩　▲同香
△3七歩　▲同香
△3六歩　▲同香
△3五歩　▲同香
探索範囲
△3四歩　▲同香

探索範囲では
角は取られない

▲3四同香まで

水平線効果：将棋

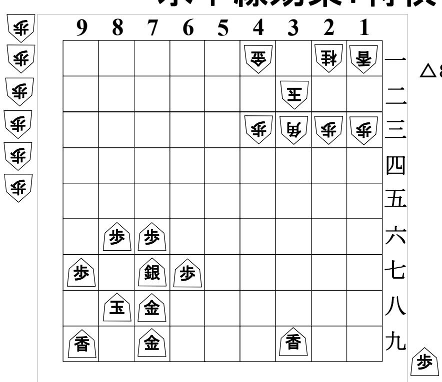


△8八歩

無意味な王手

△8八歩まで

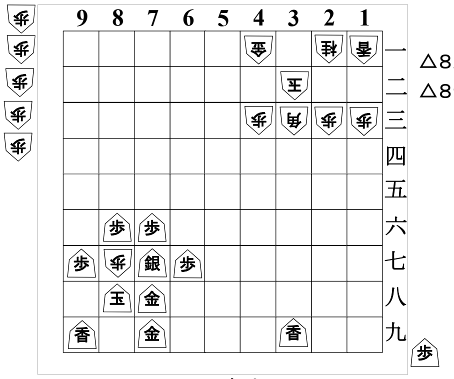
水平線効果：将棋



△8八歩　▲同玉

▲8八同玉まで

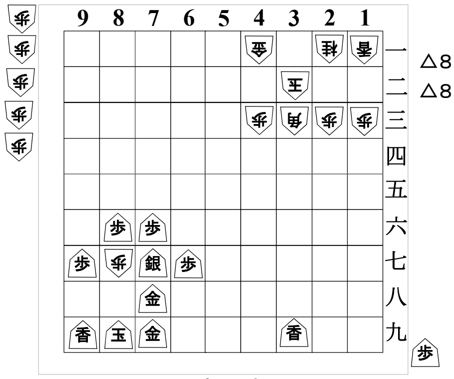
水平線効果：将棋



△8八歩　▲同玉
△8七歩

△8七歩まで

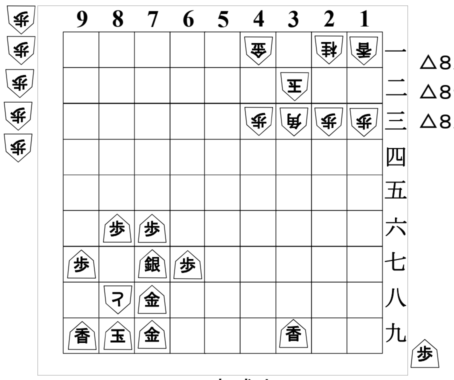
水平線効果：将棋



△8八歩　▲同玉
△8七歩　▲8九玉

▲8九玉まで

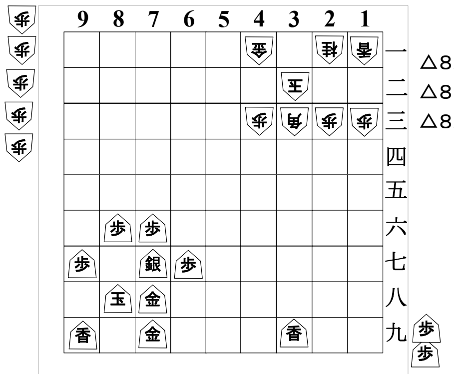
水平線効果：将棋



△8八歩　▲同玉
△8七歩　▲8九玉
△8八歩成

△8八歩成まで

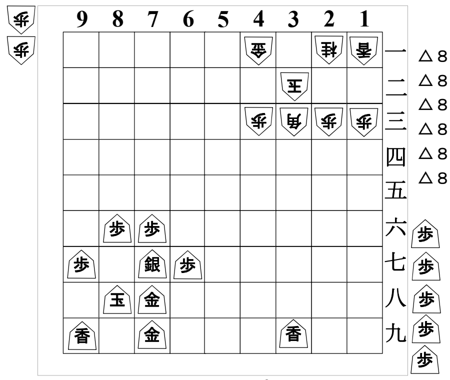
水平線効果：将棋



△8八歩　▲同玉
△8七歩　▲8九玉
△8八歩成　▲同玉

▲8八同玉まで

水平線効果：将棋



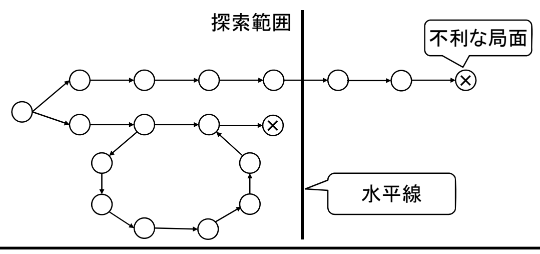
△８八歩　▲同玉
△８七歩　▲８九玉
△８八歩成　▲同玉
△８七歩　▲８九玉
△８八歩成　▲同玉
△８七歩　▲８九玉
：

歩がある限り
角を取られるのを
先延ばしにできる

▲8八同玉まで

水平線効果による問題点

- 水平線効果による問題点
  - 探索範囲外に不利な局面があっても分からない
  - 無意味な手で不利な局面を先延ばしにしてしまう



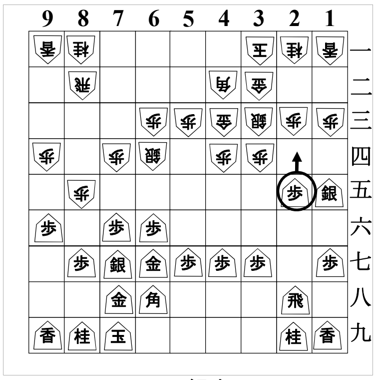
水平線効果への対処

- 水平線効果への対処
  - 水平線効果が起きそう
    ⇒先読み数を増やす

- 駒の取り合いが続いている
  ⇒取り合いが収まるまで読む
- 手を進めるにつれ評価値が徐々に下がってきている
  ⇒無意味な駒捨てをしていないかチェックする

ただしこれだけでは完全には対処できない

水平線効果への対処

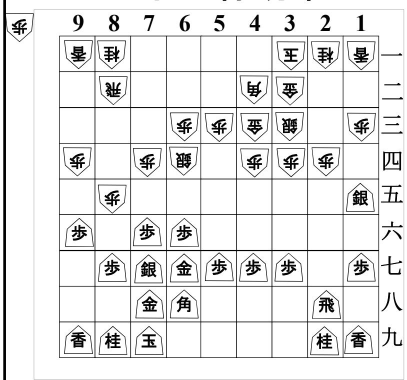


この局面で
▲2四歩は
有効？

駒の取り合いが
続く限り
読み進める

△6五銀まで

水平線効果への対処



▲2四歩　△同歩

先手の歩損

△2四同歩まで

水平線効果への対処

▲2四歩　△同歩
▲同銀　△同銀

先手の銀損

△2四同銀まで

水平線効果への対処

▲2四歩　△同歩
▲同銀　△同銀
▲同飛　△同角

取り合い終了

先手の飛車損
⇒▲2四歩は
指してはいけない

△2四同角まで

水平線効果への対処

この局面なら
▲2四歩は
有効？

△7三桂まで

水平線効果への対処

▲2四歩　△同歩
▲同銀　△同銀

先手の銀損

△2四同銀まで

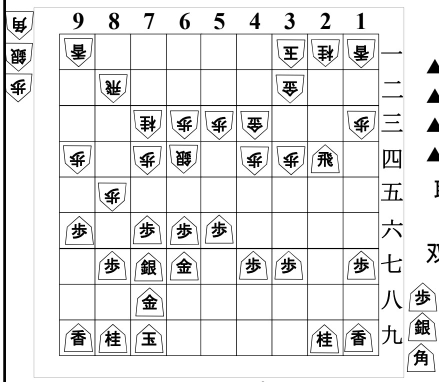
水平線効果への対処

▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角

先手の角損

△2四同角まで

水平線効果への対処



▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角
▲同飛

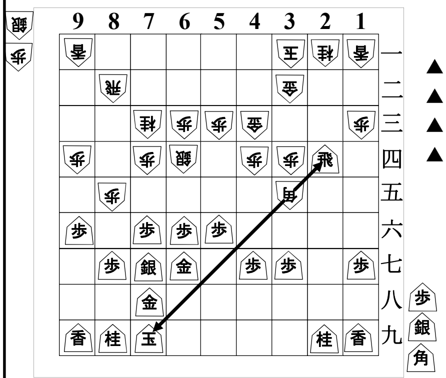
取り合い終了

双方駒損は無し
⇒▲2四歩は有効

しかし…

▲2四同飛まで

水平線効果への対処

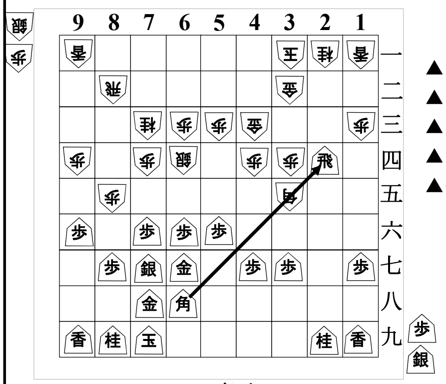


▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角
▲同飛　△3五角

△3五角で
王手飛車！

△3五角まで

水平線効果への対処



▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角
▲同飛　△3五角
▲6八角

▲6八角で
防ぐが…

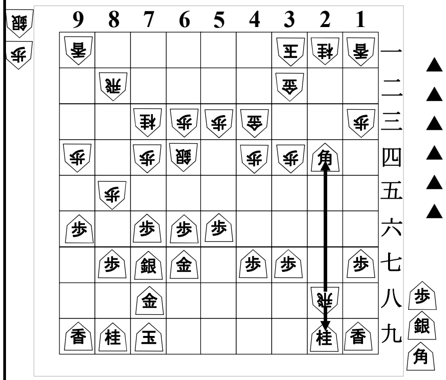
▲6八角まで

水平線効果への対処

▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角
▲同飛　△3五角
▲6八角　△2四角
▲同角

▲2四同角まで

水平線効果への対処



▲2四歩　△同歩
▲同銀　△同銀
▲同角　△同角
▲同飛　△3五角
▲6八角　△2四角
▲同角　△2八飛

角桂両取り
＋龍を作られる

⇒▲2四歩は
指してはいけない

△2八飛まで

水平線効果への対処

- 水平線効果への対処
  - 駒の取り合いが収まるまで読む
  - 無意味な駒捨てをしていないかチェックする

ただしこれだけでは完全には対処できない

現時点で水平線効果を確実に防ぐ方法は無い

課題

- 以下のテーマから1つ選び調査してください
  - 12月21日(水) 2限 発表 (5分～10分)
  - 1月11日(水) 17:00 報告書提出
    - チェス・将棋・囲碁等の強いソフト
    - チェス・将棋・囲碁等の着手選択法
    - コンピュータチェス・将棋・囲碁の歴史
    - 完全解析されているゲーム
    - 並列計算機にはどのようなものがあるか
    - LANを用いた仮想計算機
    - クラスタ処理・グリッド処理
    - その他