89

ゲーム理論
(第5回 ゲーム木探索 I)

九州大学大学院システム情報科学研究院
情報学部門
横尾 真
E-mail: yokoo@inf.kyushu-u.ac.jp
http://agent.inf.kyushu-u.ac.jp/~yokoo/

90

講義予定

4/12: イントロダクション
4/19: ゲーム理論の基礎 (I)
4/26:ゲーム理論の基礎 (II)
5/10: アドバンスドトピック
5/17: ゲーム木探索 (I)
5/24: 休講
5/31: ゲーム木探索 (II)
6/7: オークションの基礎 (I)
6/14: オークションの基礎 (II)
6/21: 組合せオークション (I)
6/28: 休講
7/5: 組合せオークション (II)
7/12: マッチング理論
7/19: 試験

91

ゲーム木探索

- 行動の選択が一回だけではなく、交互に繰り返し生じる
- 前の番に相手の選んだ手は分かる

92

例題

- 二人で交代に，1から順に25までの数を言う.
- 言う数の個数は，1個，2個，3個のいずれか好きなのを選んでよい
- 最後に25を言った方が負け

93

必勝法

- 24を言って，相手に順番を回せば絶対勝ち
- 一方，20を言って，相手に順番を回せば，相手が何個を選んでも，次に24を言える --- 絶対勝ち
- 同様に，16を言って，相手に回せば次に20を言える --- 絶対勝ち
- 同様に，12, 8, 4を言って回せば勝ち
- 先手が何を言おうと，後手は4を言って回せる
- 結局，後手が必勝

94

このゲームの性質

- 二人で交代に順番が回ってくる
- 自分の前の相手の行動／手は完全に観測できる
- 偶然の入る余地がない
- 多くのゲームは同様な性質を持つ
  - チェス，将棋，オセロ，囲碁，五目並べ，etc.
- 上記の性質を満たさないもの
  - バックギャモン：さいころ
  - ポーカー ：相手の手は見えない
  - ブリッジ:プレイヤの協調

95

必勝法

- 二人，完全情報，決定的なゲームは，原理的には必勝法が存在する
  - 先手必勝／後手必勝／引き分け
- 先手／後手を決めた時点で勝負はついている (ゲームをするまでもない)!

96

必勝法 (続き)

- 簡単なゲームなら必勝法が分かる
  - ○×（三目並べ）　引き分け
  - 五目並べ　先手必勝
  - 6x6 オセロ　後手必勝
- 複雑なゲームでは分かっていない
  - 分かってしまえばゲームは終り?

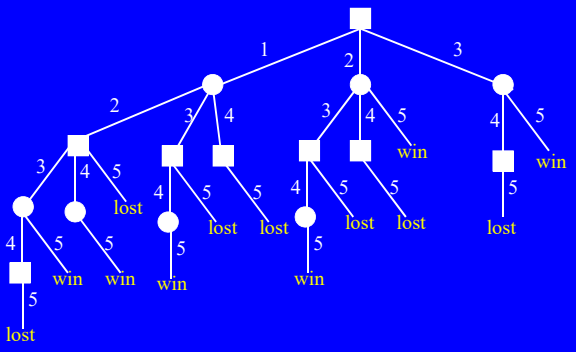
97

ゲームの木

- 状態／ノード：ゲームの可能な状態
- 状態の遷移／リンク：正しい手により遷移可能な状態間を結ぶ（一方向）.
- 先手をMAXプレイヤ，後手をMINプレイヤ，先手の順番（手番）に対応する状態をMAXノード，後手の手番の状態をMINノードと呼ぶ.
- 勝ち負けが決まったノードを端点と呼ぶ

98

例：5を言ったら負け



99

ノードのラベル付け（考え方）

- お互いに自分が勝つようにベストを尽くす
- win/lostのラベルは先手（MAXプレイヤ）の立場
- MAXプレイヤは，絶対勝てる手があればそれを選び，後手（MINプレイヤ）は，MAXプレイヤを絶対負かすことができる手があれば，それを選ぶ

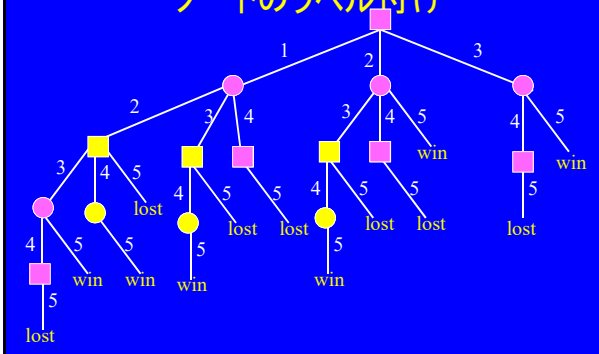
100

ノードのラベル付け

- 以下のように再帰的に定義
  - 端点に関して，そのままwin/lost
  - MAXノードに関しては，子ノードに少なくとも一つwinがあればwin，すべてlostならlost
  - MINノードに関して，子ノードに少なくとも一つlostがあればlost，すべてwinならwin
  - winを100，lostを-100とすると，上記の処理はMAXノードでは子ノードの最大値，MINノードでは最小値を取ることに対応

101

ノードのラベル付け



102

例題： ニム（コイン取り）

- コインが1個と6個の列
- 交互に，1個もしくは隣り合う2個を取る
- 最後に1個もしくは隣り合う2個を取った方が勝ち
- 先手必勝／必負?，木を書いて確かめよう

103

状態／ノード

- 各列の個数の (小さい順に並べた) リストで表現：初期状態は (1, 6)
- 初期状態から遷移可能な状態:
  (6), (1, 5), (1, 4), (1, 1, 4), (1, 2, 3)…
- すべての木を展開するのは大変なので，とりあえず (1, 4) から木を展開してみよう

104

ゲーム木の展開

必勝法を見つけるためには

- 必ずしも木を完全に展開する必要はない
  - あるMAXノードに関して，子ノードに少なくとも一つのWINがあれば，そのMAXノードはWIN
    - 他の子ノードは展開しなくても良い
  - あるMINノードに関して，子ノードに少なくとも一つLOSTがあれば，そのMINノードはLOST
    - 他の子ノードは展開しなくて良い

105

ゲーム木のサイズ

- チェッカー　10の30乗　世界チャンピオン
- オセロ　10の60乗　世界チャンピオン
- チェス　10の120乗　世界チャンピオン
- 将棋　：**2013年A級プロ棋士に勝利！**
- 囲碁　10の360　**2016年アルファ碁がトッププロに勝利！**

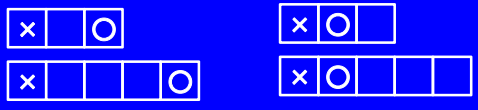
~~チェッカーでも必勝法はま~~

2007年に引き分けであることが証明された

106

例題

- 先手は○，後手は×
- 上段か下段のどちらか片方の自分の駒を動かす．左右どちらでも，いくつでも動かすことができるが，相手の駒を飛び越すことはできない．
- 自分の番で動かせないと負け
- このゲームは先手必勝／後手必勝?



107

ゲーム木が大きすぎる場合

- 普通のゲームでは，端点まで木を展開するのは不可能
- 途中まで展開されたゲーム木で，どの手が良いかを選ぶ必要がある（一手，二手，三手先まで読む等）

108

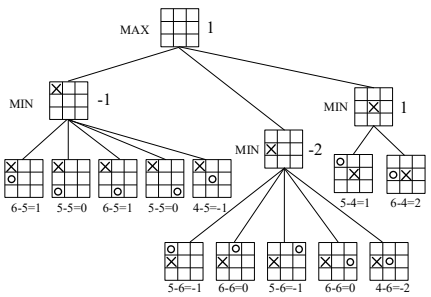
ゲーム木の評価 (MIN-MAX法)

- 途中の状態に関して，その良さを評価する関数を作る (静的評価関数)
- 評価関数は数値を返す（大きいほうが良い）
  - チェス／将棋：所有するコマの数／価値，配置等
  - オセロ:コマの数，位置 (4スミ，端)
- (ゲームが終了している訳ではない)端点の評価値を，静的評価関数の値とする
- 他のノードの評価値を，必勝法を決める方法と同様にして決める (MAXノードは最大値，MINノードは最小値)
- ルートのMAXノードで，最大値を与える経路を選ぶ.

109

静的評価関数の例

- tic-tac-too (三目並べ)で，まだ自分が取れる可能性のある列の数 − 相手が取れる可能性のある列の数



110

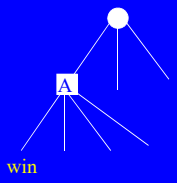
MIM-MAX探索の高速化

- 分岐をb，深さをdとすると，$O(b^d)$ のノードを展開する
- しかし，良く考えると，深さdまでのすべてのノードを展開する必要は必ずしもない

111

高速化 (I)

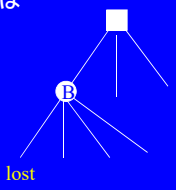
- ノードAに行ったらMAXプレイヤの勝ち
- MINプレイヤはAに行く手は選ばない
- Aの他の子ノードは展開する必要はない



112

高速化(II)

- Bに行くと，MINプレイヤの勝ち
- MAXプレイヤはBに行くパスは選ばない
- Bの他の子ノードは展開する必要がない



113

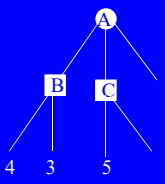
高速化手法の一般化 (アルファ・ベータ探索)

- 各ノードの評価値の下界値(それ未満には絶対ならない値)，上界値(それより大きくは絶対ならない値)を管理する
- 下界値をα値，上界値をβ値と呼ぶ
- 親がMAX, 子がMINの場合：
  - 子ノードの評価値が親の下界値以下となることが分かったら，その子ノードに関する探索は打ち切って良い
    - 親はMAX, この子ノードは選ばれない
- 親がMIN, 子がMAXの場合：
  - 子ノードの評価値が親の上界値以上になることが分かったら，子ノードに関する探索は打ち切って良い
  - 親はMIN, この子ノードは選ばれない

114

具体例 (βカット)

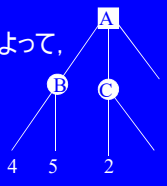
- Bの評価値が4であることが分かった時点で，MINノードAに関して，αβ(A)=(－∞, 4)
- Cの一つの子ノードの評価値が5，よって，αβ(C)＝ (5, 4)
- 5>4なので，Cに関する探索は打ち切ってよい



115

具体例(αカット)

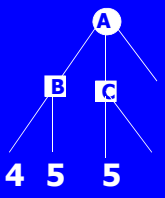
- Bの評価値が4であることが分かった時点で，MAXノードAに関して，αβ(A)=(4,∞)
- Cの一つの子ノードの評価値が2，よって，αβ(C)＝ (4, 2)
- 2<4なので，Cに関する探索は打ち切ってよい



116

具体例(深いβカット)

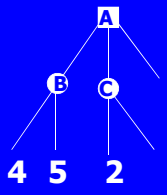
- αβ(A)=(－∞，3) であったとする
  - 3はAの祖先から得られた情報
- Bの最初の子供をチェックした時点でαβ(B)=(4, 3)
- ここでBに関する探索は打ち切られる



117

具体例(深いαカット)

- αβ(A)=(5, ＋∞) であったとする
  - 5はAの祖先から得られた情報
- Bの最初の子供をチェックした時点でαβ(B)=(5, 4)
- ここでBに関する探索は打ち切られる



118

具体的なアルゴリズム

- 関数 Vmax(n, α, β)を定義する．nはノード，α，βはそれぞれ下界値，上界値，この関数はノードnの評価値を返す．
- ルートのMAXノード r に関して，Vmax(r, －∞, ＋∞) を実行するとnの評価値が得られる．

119

関数の（再帰的な）定義

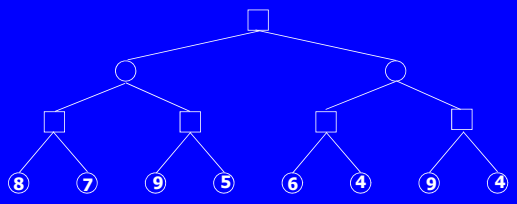
Vmax(n, α, β)
1. If n が端点 then return 静的評価関数の値
else set n_k for each child n_1, n_2, ..., n_b
2. set α=max(α, Vmin(n_k, α, β))
3. If α≧β, return β
4. If k=b, return α, otherwise, goto step 2.

Vmin(n, α, β)
1. If n が端点 then return 静的評価関数の値
else set n_k for each child n_1, n_2, ..., n_b
2. set β=min(β, Vmax(n_k, α, β))
3. If β≦α, return α
4. If k=b, return β, otherwise, goto step 2.

120

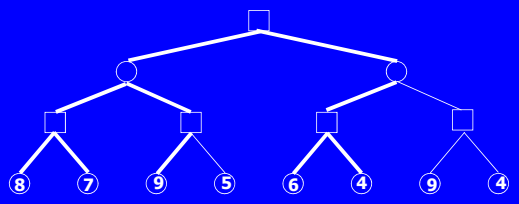
例題: アルファ・ベータ探索

- アルファ・ベータ探索で探索されるノードはどれか?



121

答え



122

アルファ・ベータ探索の効果

- ノードを展開する順序に依存する
- MAXノードに関しては，なるべく大きな評価値となる子ノード，MINノードに関しては，なるべく小さな評価値となる子ノードから展開する方が良い
- 運が悪いと展開するノード数はミニマックス探索と同じ（ぜんぜん枝刈りができない）

123

アルファ・ベータ探索の効果

- 運が良ければ，深さd，分岐bとして，探索される端点の個数は
  - MIN-MAX: b^d
  - アルファ・ベータ: $2b^{d/2}-1$
- 効果は莫大であることに注意
  - b=2として，
    - $2^2=4$
    - $2^4=16$
    - $2^8=256$
    - $2^{16}=65536$
    - $2^{32}=4.29\times10^9$