Jogos

Jerusa Marchi

jerusa.marchi@ufsc.br

Inteligência Artificial

Departamento de Informática e Estatística

UFSC

Problemas de Busca

- Os problemas de busca podem ser divididos em duas categorias principais:
  - solução de problemas
  - jogos
- A diferença está:
  - na capacidade de controle do processo (uma vez que em jogos é necessário levar em conta as jogadas do oponente)
  - no tipo de solução (em jogos a solução é parcial, no sentido de que indica qual é a melhor jogada para a situação atual)

Técnicas Utilizadas

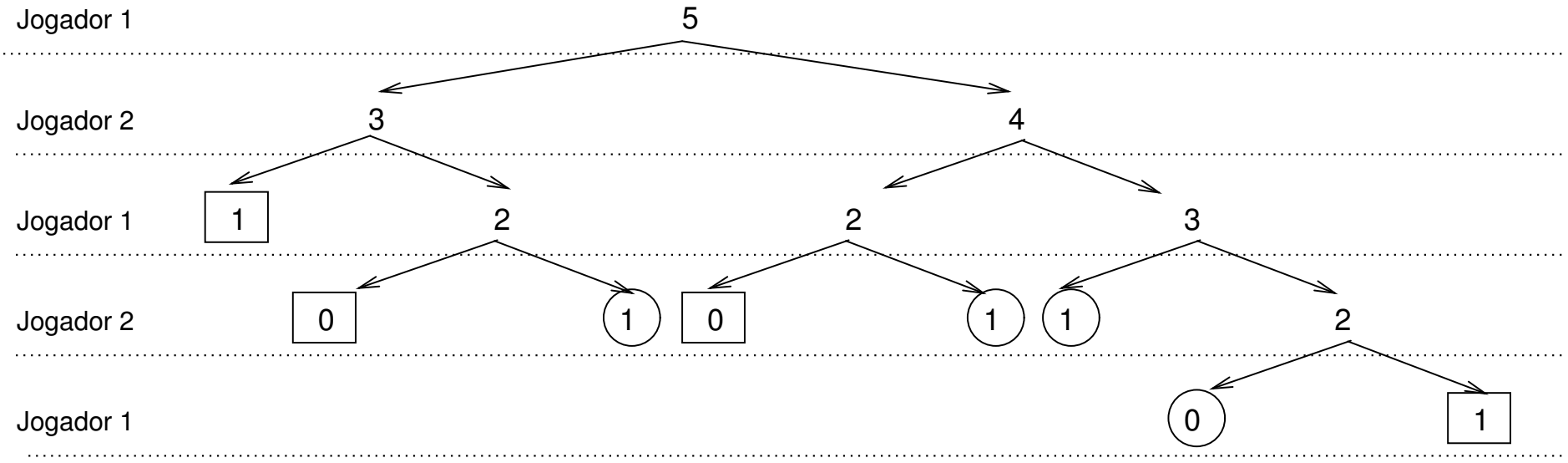
- Algoritmos MINIMAX
- Cortes Alfa e Beta
- Outros refinamentos

As técnicas utilizadas em jogos podem ser generalizadas para qualquer tipo de problema, onde, assim como em um jogo, o controle de determinadas situações não está sob o domínio do solucionador.

Jogos

- Considere o **jogo das moedas**: 5 moedas são colocadas em uma pilha e dois jogadores retiram, alternadamente, uma ou duas moedas. O jogador que retirar a última moeda é derrotado.
  - O espaço de estados contém 6 elementos (pilhas com 0, 1, 2, 3, 4 e 5 moedas)
  - Há dois operadores (retirar 1 moeda ou retirar 2 moedas)

Jogos

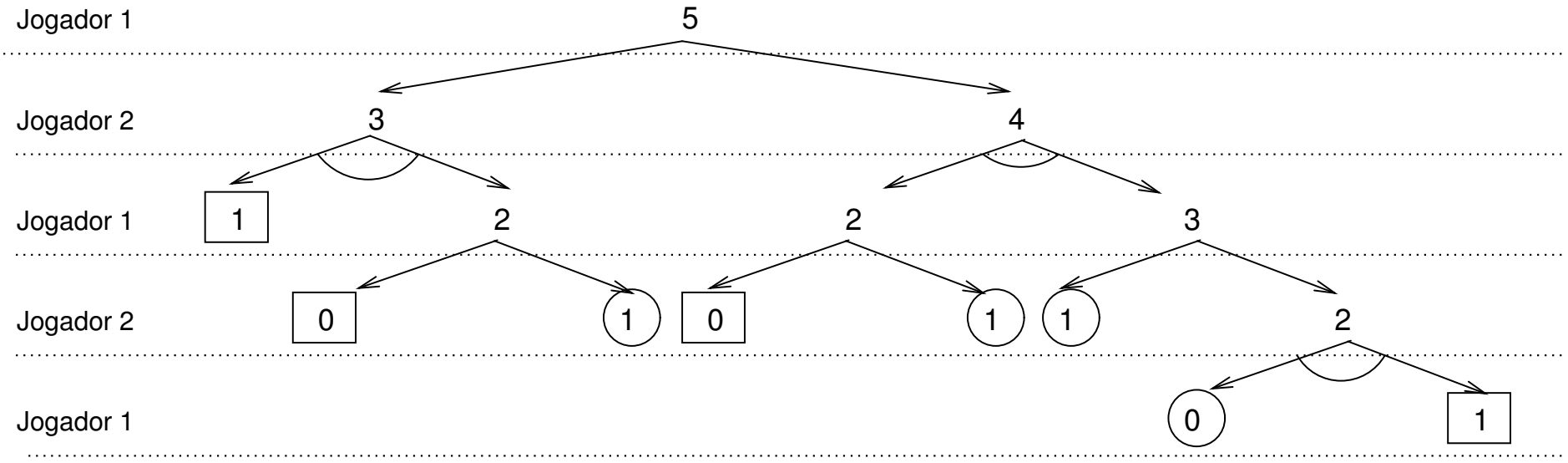


- Círculo representa vitória do jogador 1 e quadrado, derrota.

Algoritmo MINIMAX

- A árvore passa a ser vista como um grafo E/OU
  - os nodos do tipo OU representam situações onde o mecanismo é livre para escolher a próxima jogada (Jogador 1)
  - os nodos do tipo E (ligados por arcos) representam as situações onde o oponente deve jogar (Jogador 2)

Jogos

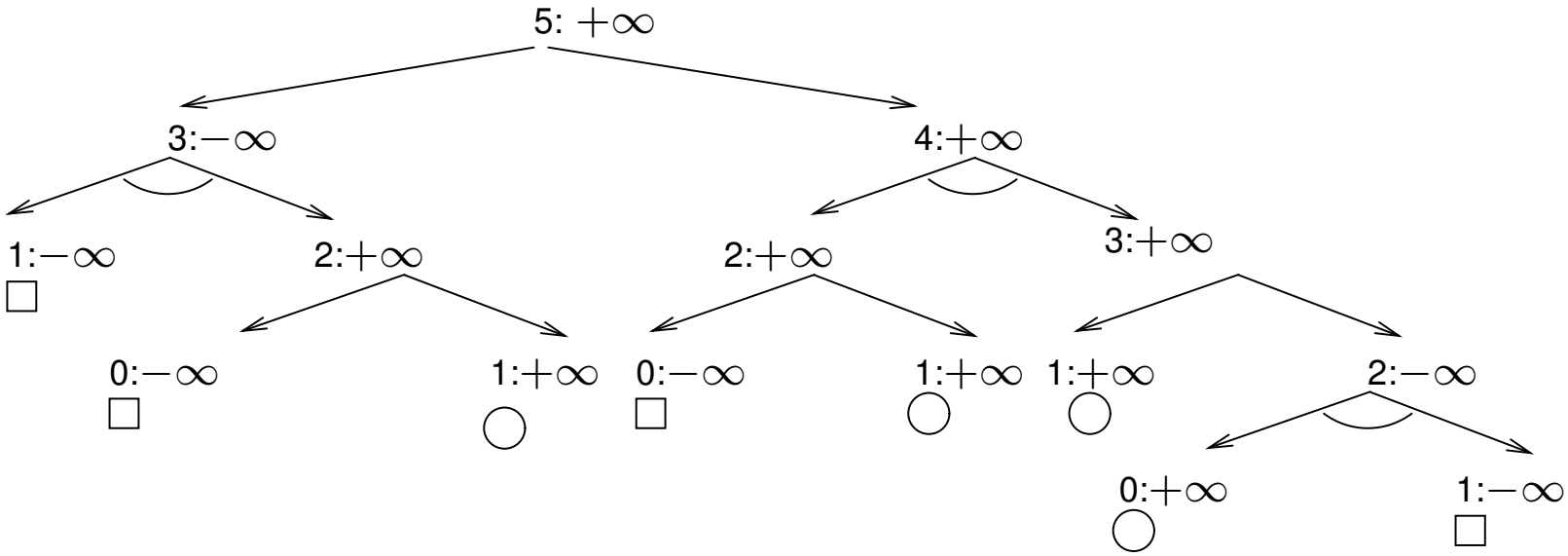


- Círculo representa vitória do jogador 1 e quadrado, derrota.

Algoritmo MINIMAX

- Funcionamento intuitivo:
  - contruir uma árvore completa, na qual a raiz corresponde à situação atual do jogo e as folhas às situações sem sucessores (vitórias, derrotas ou empates)
  - atribuir uma *avaliação estática* para cada situação sem sucessores (em geral, usa-se uma função simétrica - $+\infty$, $-\infty$, 0
  - propagar estas avalições em direção à raiz, respeitando a liberdade do oponente de escolher sua jogada:
    - a cada nodo do tipo OU, com sucessores já avaliados, atribui-se o valor *máximo* dentre as avaliações dos sucessores
    - para nodos do tipo E, atribui-se o valor *mínimo* das avaliações dos sucessores - a suposição é que o oponente sempre escolha a jogada mais proveitosa

Algoritmo MINIMAX



- Círculo representa vitória do jogador 1 e quadrado, derrota.

Algoritmo MINIMAX

- Do modo descrito, o procedimento trata nodos do tipo “E” e do tipo “OU” de modos distintos
- Simetria do jogo: regras são iguais para ambos os jogadores
- Simetria da função de avaliação estática
  - tratar todos os nodos da mesma maneira
  - a diferença entre os níveis é optida pela inversão do sinal (chamada de NEGMAX)
- para o nosso exemplo

$$fae(s) = \begin{cases} -100 \text{ se } 1 \\ +100 \text{ se } 0 \end{cases}$$

Algoritmo MINIMAX

Jogo$(estado)$

1. $suc \leftarrow Sucessores(estado)$
   $melhor \leftarrow -\infty$
   $melhor\_caminho \leftarrow \perp$
2. se $suc = \emptyset$ então retorne $(fae(estado), \perp)$
3. $\forall s \in suc$
   $resultado \leftarrow$ Jogo(s)
   seja $resultado = (valor, caminho)$
   se $-valor > melhor$
   então $melhor \leftarrow -valor$
   $melhor\_caminho \leftarrow s \oplus caminho$
4. retorne $(melhor, melhor\_caminho)$

Algoritmo MINIMAX

- Como proposto, o MINIMAX tem como desvantagem a exigência de um grafo completo, o que para a maioria dos jogos não é factível
- Uma solução é fixar uma determinada profundidade para a busca e estender a função de avaliação estática de modo que qualquer situação possa ser avaliada, e não apenas situações de vitória, empate ou derrota

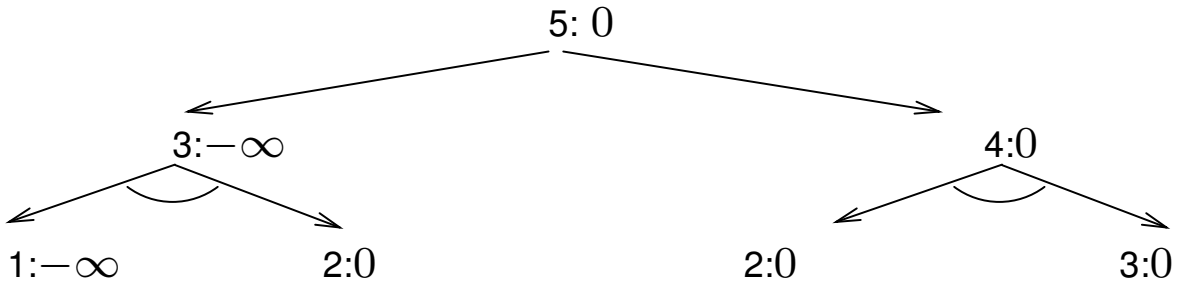
Algoritmo MINIMAX com limite

Jogo(estado, profundidade)

1. se $profundidade = limite$ então retorne $(fae(estado), \perp)$
2. $suc \leftarrow Sucessores(estado)$
   $melhor \leftarrow -\infty$
   $melhor\_caminho \leftarrow \perp$
3. se $suc = \emptyset$ então retorne $(fae(estado), \perp)$
4. $\forall s \in suc$
   $resultado \leftarrow Jogo(s, profundidade + 1)$
   seja $resultado = (valor, caminho)$
   se $-valor > melhor$
   então $melhor \leftarrow -valor$
   $melhor\_caminho \leftarrow s \oplus caminho$
5. retorne $(melhor, melhor\_caminho)$

Algoritmo MINIMAX com limite

- Usando a mesma função de avaliação estática definida anteriormente, que volta 0 caso a situação não seja de derrota ou vitória, a árvore com limite 2 é a seguinte:



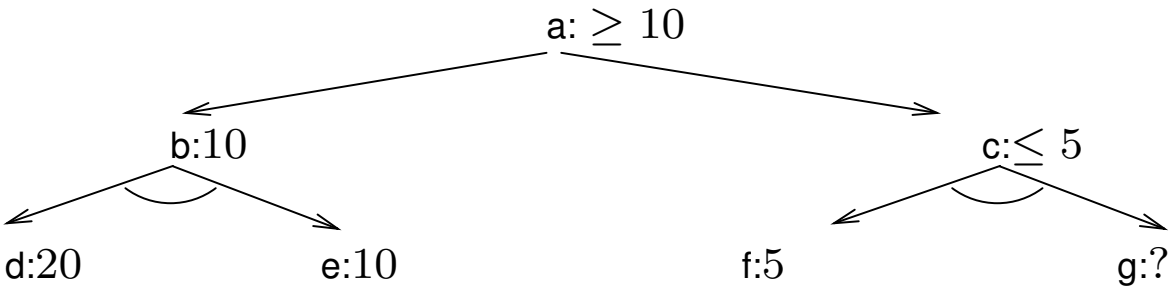
Algoritmo MINIMAX com limite

- O sucesso do algoritmo MINIMAX com limite depende diretamente da qualidade da função de avaliação estática
  - A escolha de uma boa função pode assegurar que o resultado obtido em uma busca parcial seja tão bom quanto o de uma busca completa
  - A função de avaliação precisa ser:
    - Correta - ou seja, capaz de avaliar melhor uma situação que realmente é melhor
    - Precisa - ou seja, capaz de escolher entre duas situações semelhantes qual é a melhor

Cortes Alfa e Beta

- O MINIMAX pode ser otimizado (ou seja, examina um número menor de nós) usando uma técnica ramificar e podar
  - soluções parciais claramente piores do que as soluções conhecidas são abandonadas
- Corte Alfa - usado no nível de maximização
- Corte Beta - usado no nível de minimização

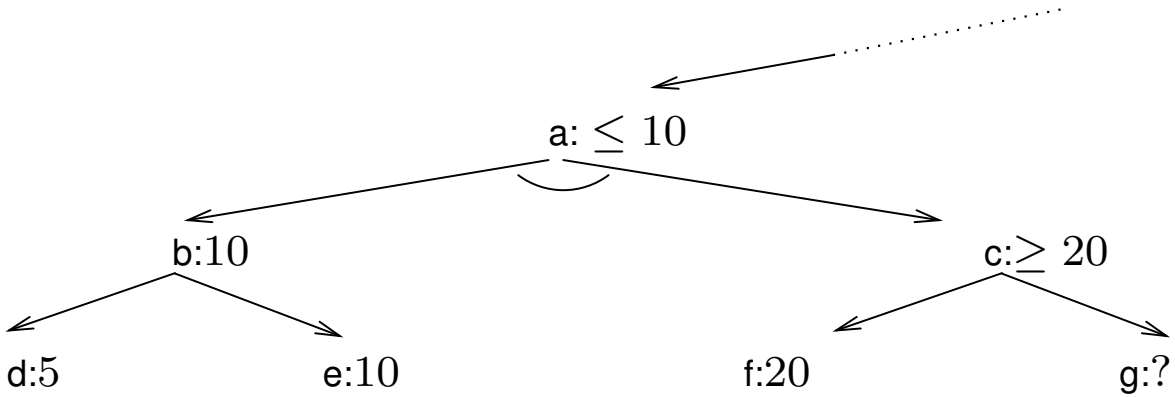
Corte Alfa



Corte Alfa

- Após percorrer o ramo da esquerda, obtendo o valor 10 como avaliação da situação “b”, pode-se concluir que o valor da situação “a”, uma vez que se trata de um nodo do tipo **OU**, será sempre maior ou igual a 10
- Se um valor menor for encontrado durante a avaliação do ramo direito, por se tratar de um nodo do tipo **E**, sabe-se que o valor atribuído ao nó da subárvore direita não poderá ser superior ao valor já encontrado
- Como do ponto de vista da situação “a” apenas interessam situações com avaliações superiores a já obtida no ramo da esquerda, a avaliação dos ramos da subárvore mais a direita (nodo g e seus filhos) torna-se irrelevante

Corte Beta



Corte Beta

- Após percorrer o ramo da esquerda, obtendo o valor 10 como avaliação da situação “b”, pode-se concluir que o valor da situação “a”, uma vez que se trata de um nodo do tipo **E**, será sempre inferior a 10
- Se um valor maior for encontrado durante a avaliação do ramo direito, por se tratar de um nodo do tipo **OU**, sabe-se que o valor atribuído ao nó da subárvore direita poderá ser somente um valor maior do que o valor já encontrado
- Como do ponto de vista da situação “a” apenas interessam situações com avaliações inferiores a já obtida no ramo da esquerda, a avaliação dos ramos da subárvore mais a direita (nodo g e seus filhos) torna-se irrelevante

Corte Alfa-Beta

- Os resultados do algoritmo MINIMAX com corte Alfa-Beta são dependentes da ordem em que os nodos são examinados
- No pior caso, o número de nodos examinados será o mesmo do procedimento MINIMAX sem a otimização

Algoritmo MINIMAX com corte Alfa-Beta

$\alpha\beta$*(estado, prof, use, passe)* /* use = maior valor fae e passe = menor valor fae*/

1. se $prof = limite$ então retorne $(fae(estado), \perp)$
2. $suc \leftarrow Sucessores(estado)$
   $melhor\_caminho \leftarrow \perp$
3. se $suc = \emptyset$ então retorne $(fae(estado), \perp)$
4. $\forall s \in suc$
   $resultado \leftarrow \alpha\beta(s, prof + 1, -passe, -use)$
   seja $resultado = (valor, caminho)$
   se $-valor > passe$
   então $passe \leftarrow -valor$
   $melhor\_caminho \leftarrow s \oplus caminho$
   se $passe \geq use$
   então retorne $(passe, s \oplus caminho)$
5. retorne $(passe, melhor\_caminho)$

Algoritmo MINIMAX com corte Alfa-Beta

- O algoritmo faz uso da simetria para tratar de modo semelhante o corte α e o corte β
- a cada nível os valores das variáveis use e $passe$ são alternados e tem seus sinais invertidos
- na primeira chamada do procedimento a variável use recebe o maior valor da função de avaliação estática e a variável $passe$ recebe o menor valor.

Refinamentos

- Caso a árvore de busca seja explicitamente armazenada em uma estrutura de dados, é possível introduzir alguns refinamentos no procedimento MINIMAX com corte Alfa-Beta, para melhorar o seu desempenho
  - Estabilidade - aplicada a jogos onde a função de avaliação estática muda bruscamente de um nível para outro da árvore (ex. damas). A ideia é continuar a busca por mais alguns níveis até que a função de avaliação estática se estabilize, evitando assim avaliações e equivocadas
  - Busca secundária - elencar as melhores jogadas obtidas no limite da profundidade e continuar, para estas, a busca por mais alguns níveis, buscando referendar a escolha das melhores jogadas
  - Jogadas de livro - usar jogadas estereotipadas para guiar o programa nestas fases do jogo

Algoritmo MINIMAX

- Considerações:
  - Mesmo com os melhoramentos e refinamentos, o algoritmo MINIMAX possui diversas limintações inerentes:
    - efeito horizonte - por mais profunda que seja a busca, primária ou secundária, a boa jogada (a jogada genial) pode estar exatamente um nível após o fim da busca, sendo perdida
    - suposição de que o adversário sempre faz a melhor jogada - esta pressuposição faz o algoritmo desenvolver um estilo de jogo conservador. Uma solução seria modelar o oponente, e de acordo com o seu desempenho, aceitar mais ou menos riscos

Curiosidade

- Garry Kasparov, campeão mundial de xadrez, após ter perdido a primeira partida para o Deep Blue, e estudando a forma como o jogador Artificial se comportava, percebeu que atacá-lo continuamente não era uma boa estratégia
- Passou então a jogar na defesa e esperar oportunidades de contra-ataque. Desta forma, empatou outras duas partidas e ganhou três
- Em 1997, a versão melhorada do Deep Blue venceu Kasparov com duas vitórias, três empates e uma derrota.

Jogos

Bibliografia:

- G. Bittencourt, *Inteligência Artificial: Ferramentas e Teorias*, 3^a Edição, Editora da UFSC, Florianópolis, SC, 2006 (cap. 4)
- E. Rich and K. Knight, *Artificial Intelligence*, McGraw-Hill, 1991 (cap. 12)