Matéria: Inteligência Artificial

Professora: Márcia

Data: 19/09/2018

Buscas locais e problemas de otimização

* Otimização: maximizar ou minimizar? Função objetivo?

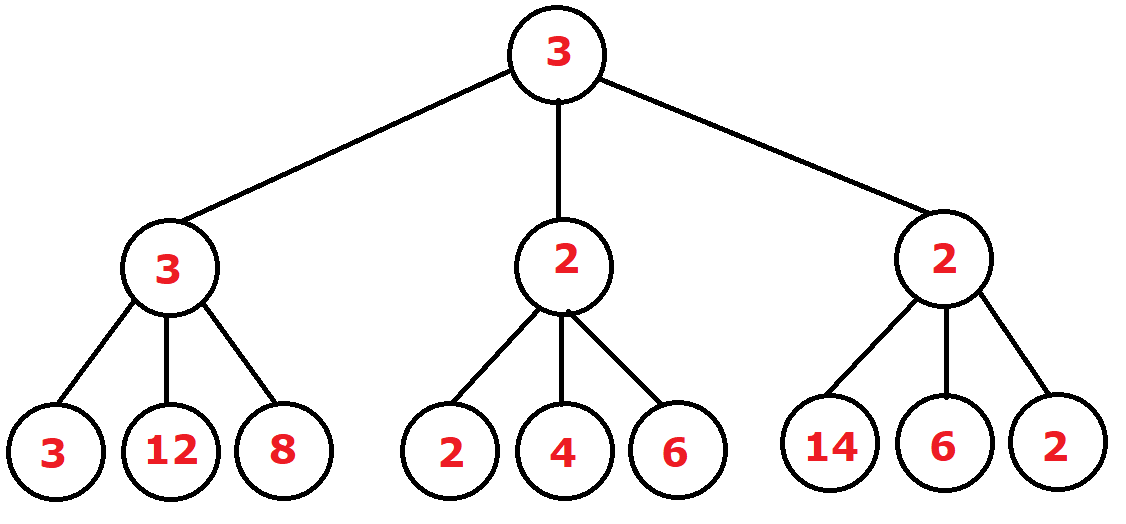
Algoritmos Genéticos

Busca Competitiva:

* Ambiente competitivo: as metas dos agentes estão em conflito.
* Estes ambiente, em IA, são ditos jogos.
* Estes ambientes são considerados determinísticos e completamente observável.
* Há dois jogadores que alternam nos movimentos.
* Cada estado é associado a um valor numérico, dito função de utilidade.
* O jogo pode ser visto como um problema de busca.
* Estado Inicial: Configuração do tabuleiro e o jogador que fará o movimento.
* Função Sucessor: Retorna uma lista de pares(movimento,estado), onde “movimento” é a “ação” que produz o “estado” resultante.
* Teste de objetivo: É o teste de término. Estados que representam o término são ditos terminais.
* Função de utilidade: Valor numérico para expressar o grau de felicidade do estado. Em geral, vitória, derrota e empate são dados por valores +1,-1,0.

Exemplo: Jogo da velha

* Estado Inicial:
  + A configuração com nove posições vazias o jogador inicial. Um jogador marca “X” e o outro “O”
* Estado de Término:
  + Estado onde há uma coluna, linha, ou diagonal completamente com o mesmo símbolo ou quando todas as posições estão completas.
* Estratégia para decisão:
  + Os dois jogadores são denominados:
    - MIN E MAX.
  + Como os jogadores se revezam, as decisões de um jogador é de acordo com a decisão de outro.
  + Para decidir, utiliza-se o valor minimax do estado corrente. Este valor depende da função de utilidade da seguinte maneira.
    - Se o estado é terminal, então o valor minimax é a função de utilidade.
    - MAX busca os valores maiores entre os valores minimax e MIN.
    - MIN busca os valores menores entre os valores minimax de MAX.
* Exemplo:



**Figura 1**

Estado Inicial:

Sucessor: Ações que podem ser executadas.

Estados Terminais:

Função de utilidade:

O cálculo do jogo é feito calculando o valor minimax de todo estado s, sabendo que aos estados terminais são atribuídas as utilidades.

minimax(n) = utilidade(n), se n é terminal

MAX minimax(s), se n é nó de MAX

s pertence sucessores(n)

MIN minimax(s), se n é nó de MIN

s pertence sucessores(n)

Algoritmo minimax(estado, -inf, +inf)

v <- max(estado, a,b)

return v;

Algoritmo max(estado,a,b)

1- if estado é terminal

2- então return utilidade(estado)

3- <- -inf

4- foreach ação a pertence s = sucessor(estado,a)

5- do v <- MAX(v,min(s),a,b)

6- return v;

if v >= b então retorna v

a <- max(a,v)

Algoritmo min(estado,a,b)

1- if estado é terminal

2- entao retorna utilidade(estado)

3- v <- +inf

4- foreach ação a pertence s = sucessor(estado,a)

5- do <- MIN(v,max(s),a,b)

6- return v

if v <= a entao retorna v

b <- min(b,v)

Árvore de jogos podem ser muito grandes, então utiliza-s a poda alfa-beta.

Há duas variáveis:

a: valor que MAX deseja maximizar

b: valor que MIN deseja minimizar