Inteligência Artificial: Busca Heurística

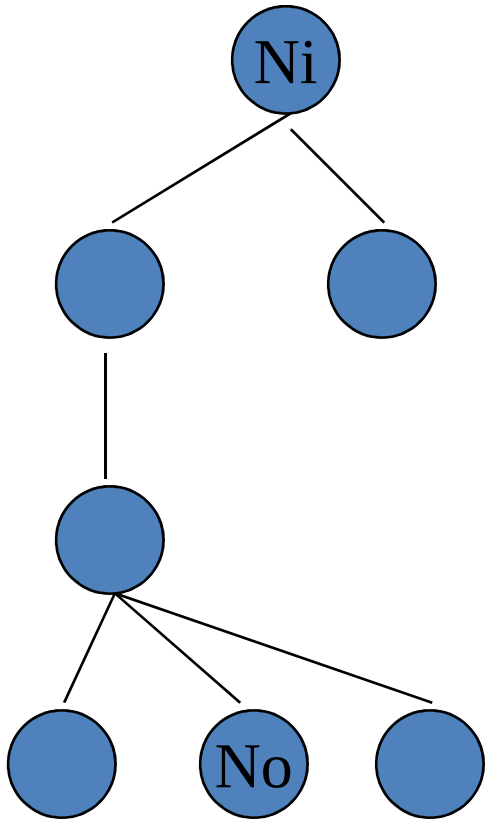
Prof. Leandro Fernandes

Em busca da solução

- Um sistema de IA pode resolver problemas da seguinte forma:
  - Ele sabe onde ele está (conjunto de informações inicial)
  - Ele sabe onde deseja ir (estado objetivo)
- Resolver problema em IA envolve busca pelo estado objetivo

Busca

- Problemas de busca são freqüentemente descritos utilizando diagramas de árvores
  - Nó inicial = onde a busca começa
  - Nó objetivo = onde ela termina
- Objetivo: Encontrar um caminho que ligue o nó inicial a um nó objetivo

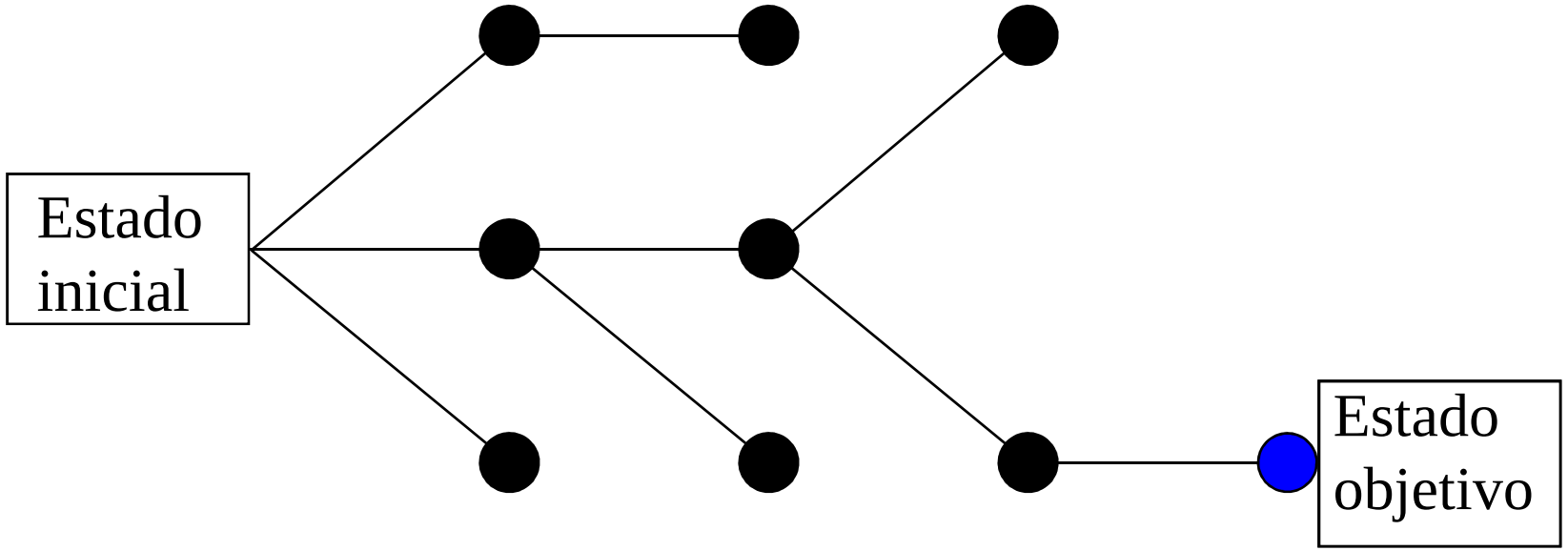


Busca

- Entrada:
  - Descrição dos nós inicial e objetivo
  - Procedimento que produz os sucessores de um nó arbitrário
- Saída:
  - Seqüência legal de nós iniciando com o nó inicial e terminando com o nó objetivo
  - Exemplo: palavras cruzadas

Uma árvore de busca

- Uma busca pode ser definida graficamente:



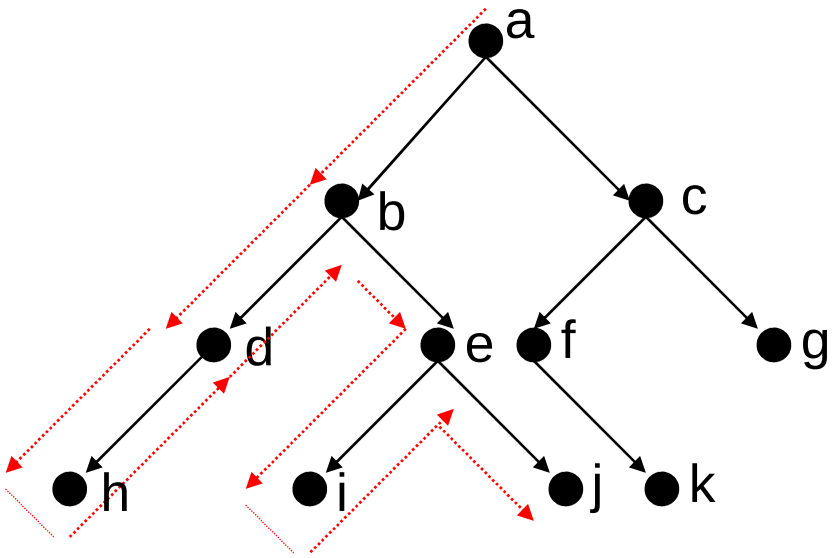
- O objetivo é atravessar a árvore partindo do estado inicial até o estado objetivo

Estratégias Básicas de Busca

- Há basicamente duas estratégias para realizarmos a busca do estado de solução em uma árvore de possibilidade, sendo estas:
  - Busca em Profundidade
  - Busca em Largura

Busca em Profundidade

- Busca pelo nó J



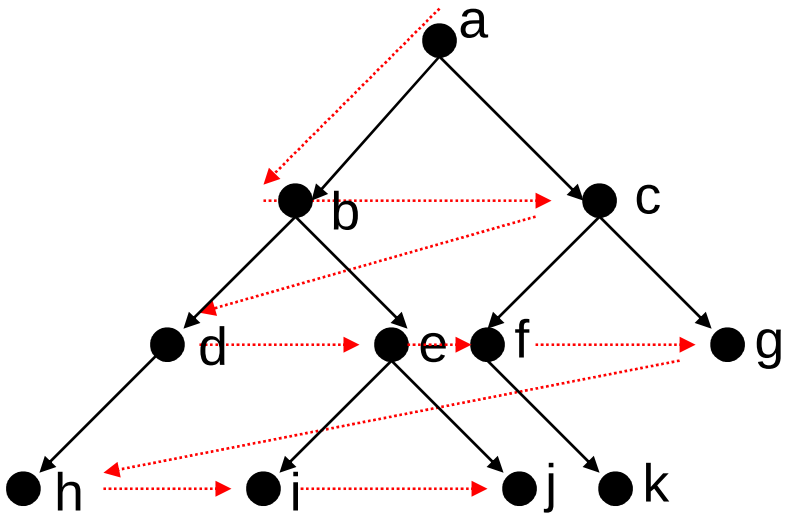
- Nó escolhido:
  - É sempre o mais ***distante*** do nó inicial

Busca em Profundidade - Algoritmo

- Se N é um nó solução então Sol = [N], ou
- Se há um nó adjacente N1 a N, tal que existe um caminho Sol1 partindo de N1 até o nó meta, então Sol = [N | Sol1]

Busca em Largura

- Busca pelo nó J



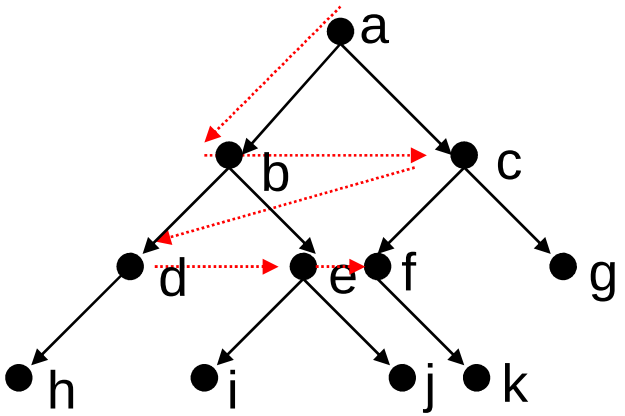
- Nó escolhido:
  - É sempre o mais ***próximo*** do nó inicial

Busca em Largura - Algoritmo

Dado um conjunto de caminhos candidatos:

- Se o primeiro caminho contém o nó meta como primeiro elemento da fila, este é uma solução, ou
- Remova o primeiro caminho do conjunto de candidatos e gere um conjunto de todas as possíveis extensões de um nó deste caminho, adicione este conjunto de extensões ao final do conjunto de candidatos e execute busca em largura no conjunto restante

Execução do Algoritmo



1 - [[a]]

2 - [[b,a] , [c,a]]

2a - [[d,b,a] , [e,b,a]]

3 - [[c,a] , [d,b,a] , [e,b,a]]

4 - [[d,b,a] , [e,b,a] , [f,c,a] , [g,c,a]]

5 - [[e,b,a] , [f,c,a] , [g,c,a] , [h,d,b,a]]

6 - [[f,c,a] , [g,c,a] , [h,d,b,a] , [i,e,b,a] , [j,e,b,a]]

Busca Heurística

- Melhores estratégias de Busca para resolução de problemas complexos

- Heurística: “conselhos” não numéricos que determinam o sucessor de um dado estado

Busca Heurística (cont)

- Possui efeito “local” - oferece um conselho de escolha do sucessor de um estado específico mas não referente à toda estratégia de busca
- Principal papel: eliminar ou podar ramos de busca

Busca Heurística (cont)

- Problema - i
  recom

Funções de Avaliação e
Funções de Custo

ecem ser

Funções de Avaliação

- Método para:
  - calcular um valor numérico para os estados sucessores de um dado estado
  - decidir pelo sucessor que tem o “melhor” valor
- Valores:
  - números não negativos
  - o menor valor é o mais promissor
  - o estado meta é 0 (zero)
- Referência ao **futuro**

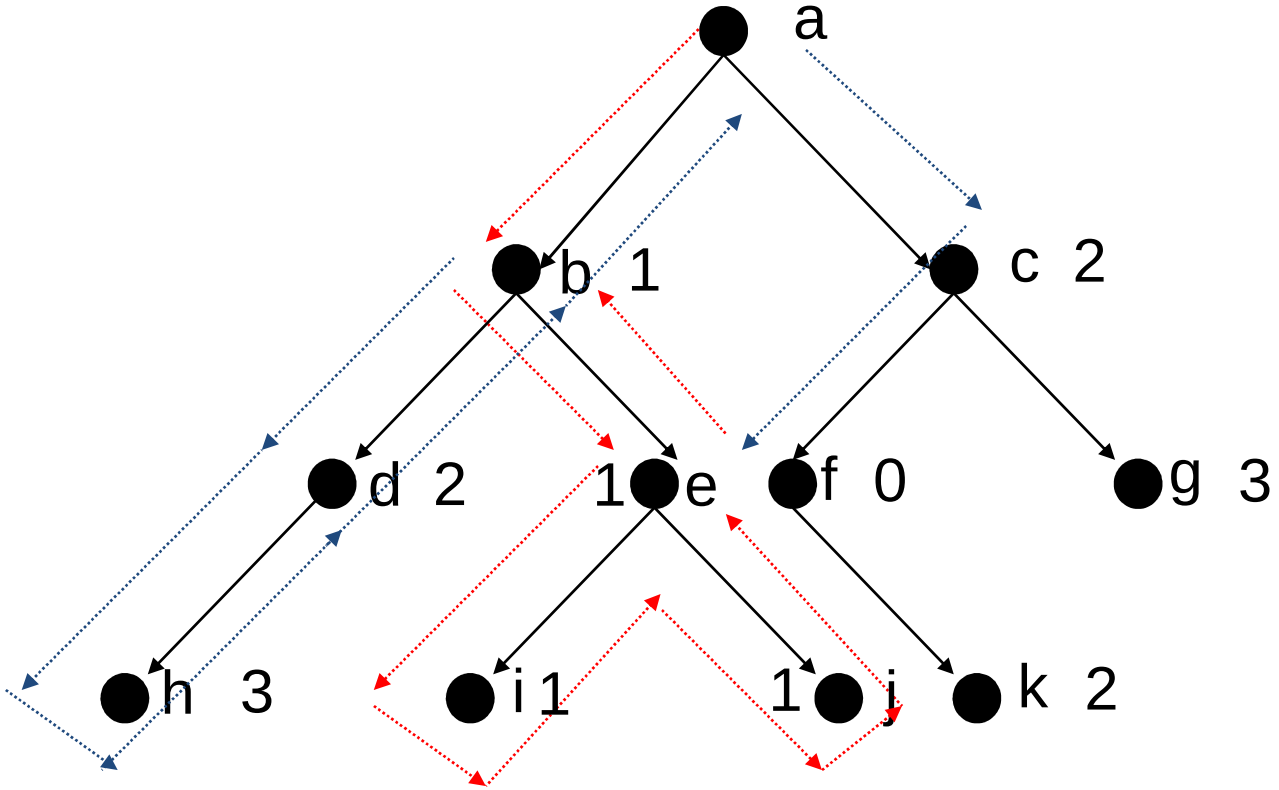
Estratégias de Busca usando Funções de Avaliação

- Hill-Climbing (ou otimização discreta)
  - consiste de uma busca em ***profundidade*** usando funções de avaliação

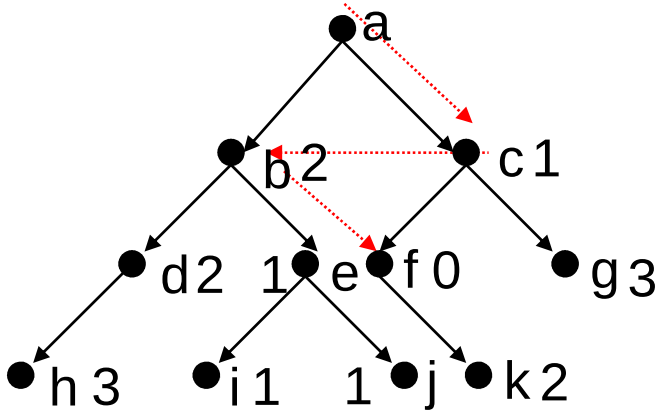
- Best-First
  - consiste de uma busca em ***largura*** usando funções de avaliação

Execução do Algoritmo Hill-Climbing

- Busca pelo no f.



Execução do Algoritmo Best-First



1 - [[a]]

2 - [[1,c,a] , [2,b,a]]

3 - [[2,b,a] , [0,f,c,a] , [3,g,c,a]]

4 - [[0,f,c,a] , [3,g,c,a] , [1,e,b,a] , [2,d,b,a]]

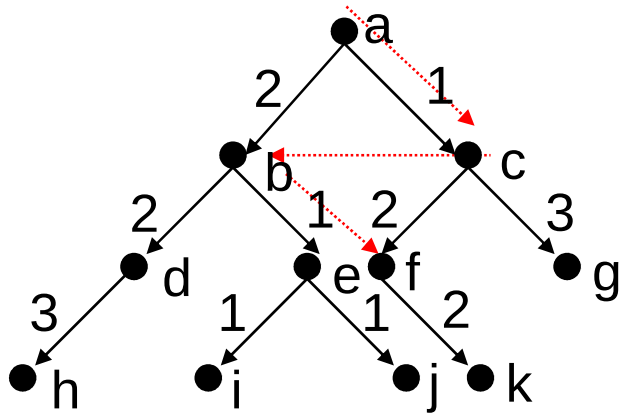
Funções de Custo

- Funções não negativas que medem a dificuldade de ir de um estado para o outro
- Usando-as, é possível encontrar um bom ou ainda o melhor caminho para alcançar uma dada meta
- Referência ao **passado**

Estratégias de Busca usando Funções de Custo

- Branch-and-Bound
  - consiste de uma busca em ***largura*** usando funções de custo

Execução do Algoritmo Branch-and-Bound



1 - [[a]]

2 - [[1,c,a] , [2,b,a]]

3 - [[2,b,a] , [3,f,c,a] , [4,g,c,a]]

4 - [[3,f,c,a] , [4,g,c,a] , [3,e,b,a] , [4,d,b,a]]

Busca Ótima (A*)

- Faz uso tanto da função de avaliação quanto da de custo
- Atribui valores de custo e avaliação aos sucessores de um estado a fim de selecionar aquele mais promissor
- Importante: os valores possuem mesmas unidades!

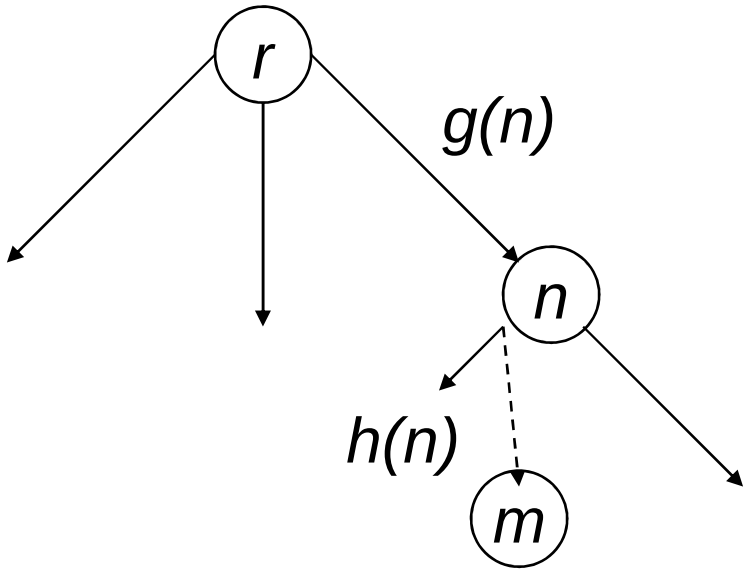
Busca Ótima (A*) (cont)

- Estimativa de custo de ir da raiz (*r*) até a meta (*m*) passando pelo nó *n:*

$$f(n) = g(n) + h(n)$$

onde:

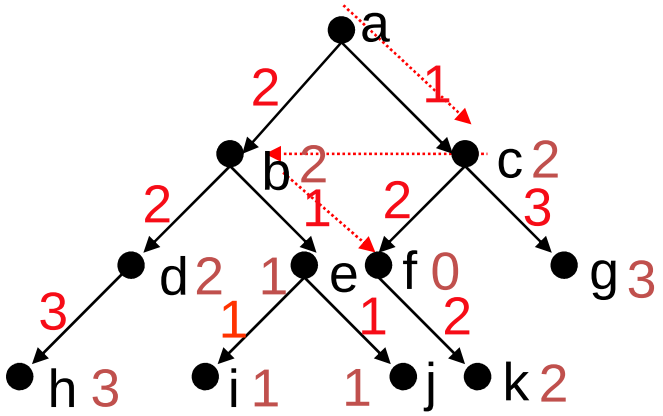
- g(n) = função de custo do nó *n*
- h(n) = função de avaliação do nó *n*



Busca Ótima (A*) (cont2)

Se a função de avaliação para qualquer estado e_i é sempre menor ou igual que o custo real de e_i para a meta,
então o primeiro caminho encontrado pela estratégia de busca A* é o caminho de custo mínimo **(ótimo)**

Execução do Algoritmo A*



1 - [[a]]

2 - [[1,2,c,a] , [2,2,b,a]]

3 - [[2,2,b,a] , [3,2,f,c,a] , [4,3,g,c,a]]

4 - [[3,2,f,c,a] , [4,3,g,c,a] , [3,1,e,b,a] , [4,2,d,b,a]]

Organização das Estratégias de Busca

Estratégias de Busca	**Usa Agenda?**	**Usa função de avaliação**	**Usa função de custo**	**Próximo estado**
Profundidade	*Não*	*Não*	*Não*	o sucessor do último estado, caso contrario o sucessor do predecessor
Largura	*Sim*	*Não*	*Não*	o estado mais longe na agenda(fila)
Hill-Climbing	*Não*	*Sim*	*Não*	o sucessor com mínimo valor de função de avaliação
Best-First	*Sim*	*Sim*	*Não*	o estado na agenda com mínimo valor de função de avaliação
Branch-and-Bound	*Sim*	*Não*	*Sim*	o estado na agenda com mínimo valor de função de custo total
A*	*Sim*	*sim*	*Sim*	o estado na agenda com mínimo valor da soma da função de avaliação e custo total