Buscas Online

Agentes que exploram ambientes desconhecidos

Prof. Marcelo de Souza

85ECS – Engenharia de Software Orientada a Agentes
Universidade do Estado de Santa Catarina

UDESC

Busca offline *vs.* busca online

Planejamento e otimização

- Algoritmos clássicos de busca (offline);
- Ambiente: completo (observável), determinístico e estático;
- Percepção → solução completa → executa solução.
  - Não recorrem a novas percepções e não replanejam a solução!

Busca offline *vs.* busca online

Planejamento e otimização

- Algoritmos clássicos de busca (offline);
- Ambiente: completo (observável), determinístico e estático;
- Percepção → solução completa → executa solução.
  - Não recorrem a novas percepções e não replanejam a solução!

Busca online

- ação → observa o ambiente → calcula a próxima ação → ...
- Ideal para ambientes:
  - Dinâmicos (ou semi-dinâmicos, onde há um tempo para deliberação);
  - Não determinísticos;
  - **Desconhecidos**: quais os estados e o que fazem as ações?

Ambientes desconhecidos

Ideia geral

- Usar ações como experimentos para **explorar** e **aprender** sobre o ambiente;
- Computação "pura" não resolve.

Ambientes desconhecidos

Ideia geral

- Usar ações como experimentos para **explorar** e **aprender** sobre o ambiente;
- Computação "pura" não resolve.

O agente conhece:

- ações(s): função que retorna a lista de ações permitidas no estado s;
- teste(s): função que testa se s é um estado objetivo.

O agente deve explorar (e aprender):

- S: o conjunto de estados do ambiente;
- $c(s, a)$: função de custo de executar a ação a no estado s;
- resultado(s, a): o efeito da ação a no estado s (i.e. estado seguinte s').

Estratégias de exploração

Busca offline

- exploração é feita antes de executar qualquer ação;
- permite explorar vários caminhos da árvore.

Busca online

- exploração: executar ações e observar os resultados;
- explorar um caminho diferente implica em "desfazer" uma série de ações (**custoso!**);
- exige estratégias de exploração específicas.

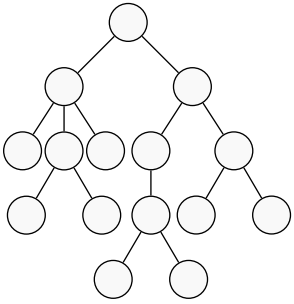
Estratégias de exploração

Busca offline

- exploração é feita antes de executar qualquer ação;
- permite explorar vários caminhos da árvore.

Busca online

- exploração: executar ações e observar os resultados;
- explorar um caminho diferente implica em "desfazer" uma série de ações (**custoso!**);
- exige estratégias de exploração específicas.



Imagine uma busca em largura na árvore ao lado.

- Busca em profundidade parece ideal para busca online!

Busca em profundidade online

Busca em profundidade online

Entrada : s' – percepção do estado atual

se *teste(s')* **então retorna** solução

se $s' \notin$ *experimentar (i.e. novo estado)* **então**
  experimentar[s'] ← ações(s')

se $s \neq$ *null* **então**
  resultado[s, a] ← s'
  empilha s a retroceder[s']

se *experimentar[s'] é vazio* **então**
  se *retroceder[s'] é vazio* **então retorna** falha
  $a \leftarrow$ ação b tal que resultado[s', b] = desempilha(retroceder[s'])
  $s' \leftarrow$ null

senão $a \leftarrow$ desempilha(experimentar[s'])

$s \leftarrow s'$

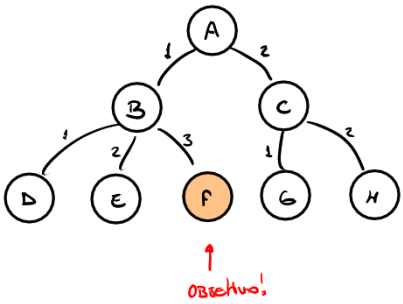
retorna a

- no início, o agente só conhece seu estado atual s' e suas ações;
- s e a são o estado e ação anteriores;
- experimentar: ações a serem exploradas;
- resultado: conhecimento do agente;
- retroceder: estado de onde veio;
- agente explora e aprende enquanto busca o objetivo.

Busca em profundidade online

Exemplo (execução)

Execute a busca em profundidade online na árvore de estados abaixo.



VARIÁVEIS

$s' = A$

$s = NULL$

$a = NULL$

RESULTADO

	A	B	C	D	E	F	G	H
1								
2								
3								
-1								
-2								
-3								

VOLTA

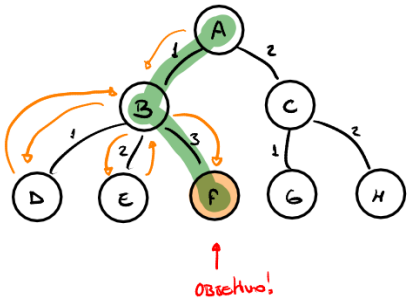
EXPERIMENTAR

RETROCEDER

Busca em profundidade online

Exemplo (execução)

Execute a busca em profundidade online na árvore de estados abaixo (**estado final**).



VARIÁVEIS

$s' = F$

$s = B$

$a = 3$

RESULTADO

	A	B	C	D	E	F	G	H
1	B	D						
2		E						
3								
-1		A		B				
-2					B			
-3								

VOLTA

EXPERIMENTAR

$A \rightarrow \cancel{B}, C$

$B \rightarrow \cancel{D}, \cancel{E}, \cancel{F}$

$D \rightarrow \emptyset$

$E \rightarrow \emptyset$

RETROCEDER

$B \rightarrow A$

$D \rightarrow B$

$E \rightarrow B$

Busca em profundidade online

Quando o agente falha?

- Quando explora todos os estados e volta ao estado inicial.
- Neste caso, ele explorou todas suas ações e não tem pra onde retroceder.

Variantes:

- Busca em profundidade **limitada** online;
- Busca em **aprofundamento iterativo** online.

Busca local online

Por definição, buscas locais (*hill climbing*) expandem estados baseados na **localidade**.

- Ou seja, o algoritmo mantém o estado atual e explora as ações nesse estado;
- *Hill climbing* já é um algoritmo de **busca online**;
- **Problema**: o algoritmo simples fica preso em ótimos locais e *plateaus*.

Busca local online

Por definição, buscas locais (*hill climbing*) expandem estados baseados na **localidade**.

- Ou seja, o algoritmo mantém o estado atual e explora as ações nesse estado;
- *Hill climbing* já é um algoritmo de **busca online**;
- **Problema**: o algoritmo simples fica preso em ótimos locais e *plateaus*.

Que tal usar reinícios aleatórios?

- Não é possível, pois o agente não pode se transportar a outro estado;
- Mas, podemos fazer uma caminhada aleatória de $\mathfrak{n}$ passos (executar ações aleatórias), o que leva o agente a um estado aleatório;
- **Problema**: o algoritmo é lento em cenários complexos.

Busca local online

Por definição, buscas locais (*hill climbing*) expandem estados baseados na **localidade**.

- Ou seja, o algoritmo mantém o estado atual e explora as ações nesse estado;
- *Hill climbing* já é um algoritmo de **busca online**;
- **Problema**: o algoritmo simples fica preso em ótimos locais e *plateaus*.

Que tal usar reinícios aleatórios?

- Não é possível, pois o agente não pode se transportar a outro estado;
- Mas, podemos fazer uma caminhada aleatória de $\mathfrak{n}$ passos (executar ações aleatórias), o que leva o agente a um estado aleatório;
- **Problema**: o algoritmo é lento em cenários complexos.

Solução: usar buscas locais com **informação**, **memória** e **aprendizado**!

Busca local online

Learning real-time A* (LRTA*)

Entrada : s' – percepção do estado atual

se *teste(s')* **então retorna** solução

se $s' \notin H$ *(i.e. novo estado)* **então**
  $H[s'] \leftarrow h(s')$

se $s \neq$ *null* **então**
  resultado$[s, a] \leftarrow s'$
  $H[s] \leftarrow \min(\text{custo}(s, b), \forall b \in \text{ações}(s))$

$a \leftarrow \text{argmin}_b(\text{custo}(s', b), \forall b \in \text{ações}(s'))$
$s \leftarrow s'$
retorna a

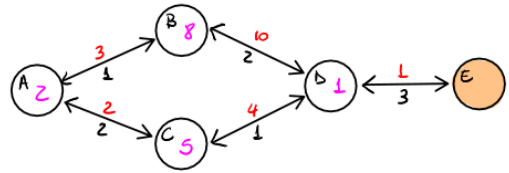
function custo(s, a):
  se *resultado[s, a] é desconhecido* **então retorna** $h(s)$
  senão retorna $H[\text{resultado}[s, a]] + c(s, a)$

- no início, o agente só conhece seu estado atual s' e suas ações;
- s e a são o estado e ação anteriores;
- $h(s)$: função heurística para um estado s;
- H: custo esperado de cada estado até o objetivo;
- resultado: conhecimento do agente;
- agente aprende a tomar decisões.

Busca local online

Exemplo (execução)

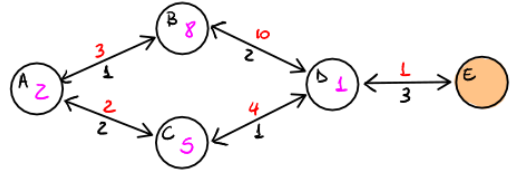
Execute a busca LRTA* no grafo de estados abaixo.



Busca local online

Exemplo (execução)

Execute a busca LRTA* no grafo de estados abaixo (**estado final**).



Buscas online

Note que:

- A medida que o agente testa ações nos estados em que se encontra, ele aprende os efeitos de suas ações. Ou seja, ele toma conhecimento da transição de estados.
- Ele também atualiza a qualidade de cada estado para tomar decisões futuras.
- Esse conhecimento pode ser armazenado para execuções futuras, onde parte do ambiente já é conhecida.

Buscas online

Note que:

- A medida que o agente testa ações nos estados em que se encontra, ele aprende os efeitos de suas ações. Ou seja, ele toma conhecimento da transição de estados.
- Ele também atualiza a qualidade de cada estado para tomar decisões futuras.
- Esse conhecimento pode ser armazenado para execuções futuras, onde parte do ambiente já é conhecida.

Para avaliar a qualidade de um agente de busca online:

- Razão competitiva: caminho feito pelo agente *vs.* caminho ótimo;
- Número de estados avaliados *vs.* tamanho do espaço de estados.

Buscas online

Note que:

- A medida que o agente testa ações nos estados em que se encontra, ele aprende os efeitos de suas ações. Ou seja, ele toma conhecimento da transição de estados.
- Ele também atualiza a qualidade de cada estado para tomar decisões futuras.
- Esse conhecimento pode ser armazenado para execuções futuras, onde parte do ambiente já é conhecida.

Para avaliar a qualidade de um agente de busca online:

- Razão competitiva: caminho feito pelo agente *vs.* caminho ótimo;
- Número de estados avaliados *vs.* tamanho do espaço de estados.

Técnicas mais sofisticadas (exigem treinamento em ambiente simulado):

- Processos de decisão de Markov;
- Aprendizado por reforço.

85ECS – Engenharia de Software Orientada a Agentes
Prof. Marcelo de Souza