

Inteligência Artificial

Prof. Fabrício Olivetti de França Prof. Denis Fantinato
3º Quadrimestre de 2019

Busca Informada

Definição de um problema:

- Conjunto de estados
- Conjunto de ações
- Função de transição
- Função de custo
- Estado inicial e estado objetivo

Árvore de Busca:

- Nós representam os possíveis estados.
- Arestas representam transições de estado.
- Um caminho em uma árvore (da raiz ao nó folha) representa uma solução.

Recapitulando

Algoritmos de busca constroem sistematicamente a árvore de busca.

Diferenciam pela ordem que constroem os nós.

Algoritmo ótimo: encontra a solução com o menor custo.

Explora as opções sem nenhuma informação de onde está o objetivo.

Expande caminhos ruins desnecessariamente.

Exemplo de Problemas

Vamos ilustrar alguns problemas fictícios e reais para entendermos melhor os conceitos da aula de hoje.

8-Puzzle

7	2	4
5		6
8	3	1

Start State

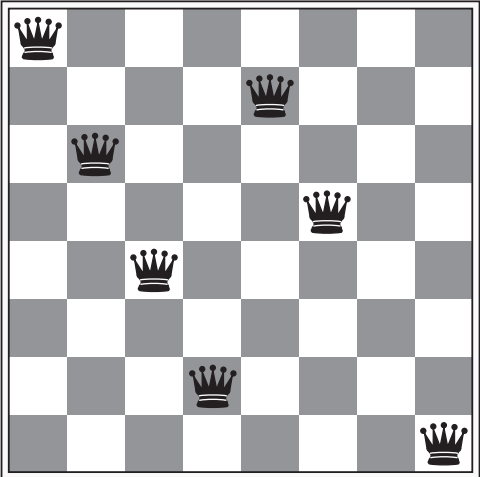
	1	2
3	4	5
6	7	8

Goal State

8-Puzzle

- **Estados:** uma matriz 3x3 cada posição contendo um número de 0 a 8, sendo o valor 0 um espaço vazio.
- **Estado inicial:** qualquer um dos estados.
- **Objetivo:** Reordenar os valores na ordem de 0 a 8 dentro da matriz.
- **Ações:** Esquerda, Direita, Cima, Baixo (ou um subconjunto desse).
- **Transição:** Dado um estado **s** e uma ação **a**, efetua-se a troca do valor 0 com o valor vizinho na direção de **a**.
- **Custo:** Cada movimento tem custo 1.

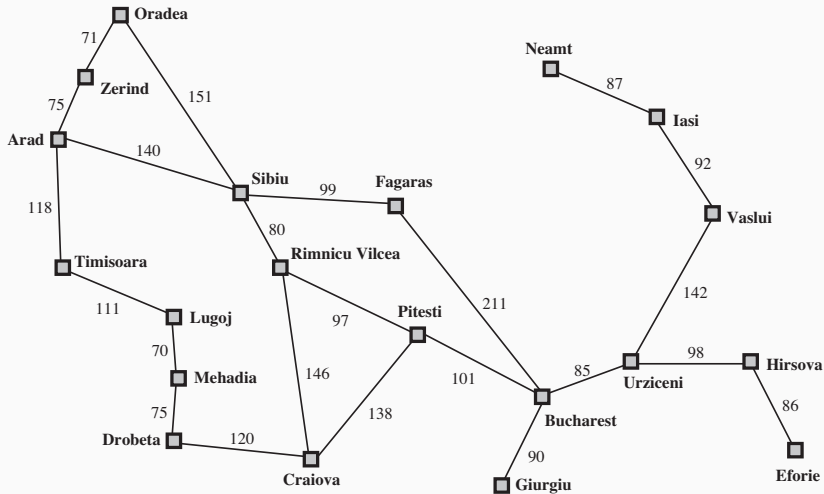
8 Rainhas



8 rainhas

- **Estados:** um arranjo qualquer de 0 a 8 rainhas no tabuleiro de xadrez.
- **Estado inicial:** qualquer um dos estados.
- **Objetivo:** posicionar as rainhas de tal forma que nenhuma é atacada.
- **Ações:** adicionar uma rainha em uma posição.
- **Transição:** dado um estado **s** e uma ação **a**, insere a rainha na posição definida em **a**.
- **Custo:** quantidade de rainhas sendo atacadas.

Menor rota



Menor rota

- **Estados:** local atual. Qualquer uma das cidades consideradas.
- **Estado inicial:** ponto de partida da rota.
- **Objetivo:** destino final da rota.
- **Ações:** escolher um estado que possui ligação com o atual.
- **Transição:** dado o estado atual s , seguir para o estado s' definido em a caso seja vizinho de s .
- **Custo:** custo de sair de s e chegar em s' .

Na Busca informada o agente faz decisões informadas sobre o caminho mais interessante para continuar a busca.

Para isso utilizamos heurísticas de forma a estimar qual o melhor caminho até o objetivo.

Heurísticas

Heurística

Do grego, *heuriskō*, significa *encontrar, descobrir* (*eureka!!*).

Na área de busca e otimização é um método que tem o objetivo de encontrar uma solução para um problema sem garantias teóricas de otimalidade.

Tenta satisfazer um objetivo imediato e intermediário.

Em problemas de busca é modelado em forma de uma função $h(n)$ que estima o custo do melhor caminho entre o estado descrito pelo nó n até o objetivo.

Para o problema do 8 puzzle podemos considerar uma função heurística que soma a a distância entre cada número e sua posição correta.

Para o problema das 8 rainhas, podemos utilizar a quantidade de quadrados não atacados por nenhuma rainha.

Para o problema da rota, uma heurística razoável seria a distância em linha reta da posição atual até o destino.

Veja que embora essas funções não garantem que o caminho a ser seguido é o melhor, eles trazem uma intuição sobre os melhores candidatos.

Busca com Heurística

Também conhecida como **Best-first search** e **Greedy search**. Expande o nó que minimiza o custo estimado pela função heurística.

Busca Gulosa

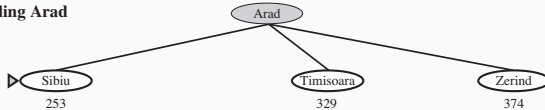
```
def buscaGulosa(nos, h):  
    no = argmin(nos, h)  
    sl = [resultado(no, a) for a in acoes(no)]  
    if any(atingiuObj(s) for s in sl):  
        return filter(atingiuObj, sl)[0]  
    nos = nos.remove(no) + sl  
  
    if empty(nos):  
        return Falha  
  
    return buscaGulosa(nos, h)
```


Busca Gulosa

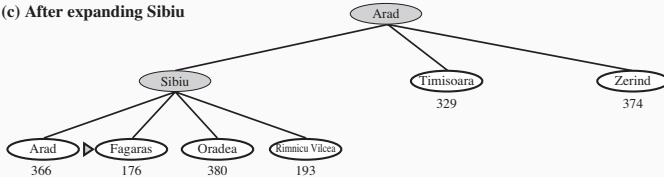
(a) The initial state



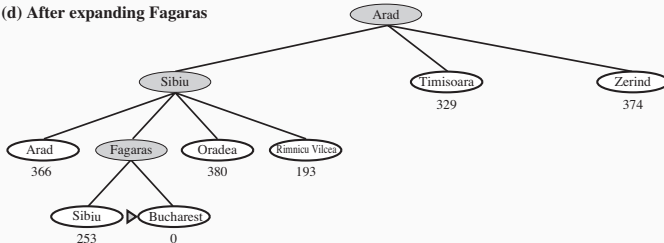
(b) After expanding Arad



(c) After expanding Sibiu



(d) After expanding Fagaras



Exercício

Aplique a Busca em Largura e a Busca Gulosa no problema 8 puzzle iniciando do estado abaixo. Compare os resultados!

7	2	4
5	0	6
8	3	1

Outra estratégia **best-first** é conhecida como A*, ou **A-estrela**.

A ideia é a mesma da busca gulosa, porém a escolha do nó a expandir agora é definida pela função:

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

Sendo $g(n)$ o custo do estado inicial até o nó n .

Dessa forma levamos em conta não apenas o custo para chegar até o objetivo, mas também o que gastamos até então.

Esse algoritmo pode garantir otimalidade em certas condições.

```
def buscaAStar(nos, g, h):  
    no = argmin(nos, g, h)  
    sl = [resultado(no, a) for a in acoes(no)]  
    if any(atingiuObj(s) for s in sl):  
        return filter(atingiuObj, sl)[0]  
    nos = nos.remove(no) + sl  
  
    if empty(nos):  
        return Falha  
  
    return buscaAStar(nos, h)
```

Heurística Admissível

Uma heurística é dita **admissível** se ela nunca superestima o custo $h(n)$ de atingir o objetivo a partir do nó n . Ela é otimista!

Por consequência, dado um $h(n)$ admissível, $f(n) = g(n) + h(n)$ também é admissível.

Heurística Admissível

A heurística do caminho mais curto é admissível pois é impossível termos uma distância menor que uma linha reta entre dois pontos.

Otimidade no A*

Dado uma heurística $h(n)$ admissível, temos a garantia de otimalidade no algoritmo A*.

Otimidade no A*

Considere o conjunto atual de nós a serem explorados é $[n1, n2]$ e que o caminho de $n1$ leva ao objetivo n^* e que $g(n1) < g(n2)$.

Com isso temos que $f(n1) \leq g(n^*)$.

Com isso temos que $f(n_1) \leq g(n^*)$ e $f(n^*) = g(n^*)$.

Otimidade no A*

Com isso temos que $f(n1) \leq f(n^*)$.

Temos também que $f(n^*) < f(n2)$, pois n^* é ótimo.

Logo, $f(n1) \leq f(n^*) < f(n2)$.

Otimidade no A*

Note que embora possamos dizer que o algoritmo A* é ótimo, não significa que ele consegue encontrar a resposta em um tempo hábil.

O espaço de busca da fronteira a ser explorada ainda cresce exponencialmente com o tamanho da melhor solução.

Quanto mais próximo é $h(n)$ do valor de custo real, mais rapidamente a solução poderá ser obtida.

8 puzzle

Dada as duas heurísticas abaixo, elas são admissíveis?

- 1) h_1 = número de valores fora de sua posição correta
- 2) h_2 = a soma das distância de cada valor para sua posição correta.

Exercício

Aplique o algoritmo A* no problema 8 puzzle partindo do estado abaixo utilizando $h1$ e $h2$. Compare os resultados:

7	2	4
5	0	6
8	3	1

Criando uma heurística

Uma forma de pensar em uma heurística é tentar criar heurísticas para *relaxações* do problema principal. Por exemplo:

- 1) Se cada movimento do 8 puzzle puder ser a troca dos valores de quaisquer duas posições, a função $h1$ é ótima.
- 2) Se cada movimento puder ser a troca de quaisquer dois valores adjacentes (e não apenas o 0), chegamos em $h2$.

Criando uma heurística

Quanto mais próximo um problema relaxado for do problema real, melhor será nossa heurística.

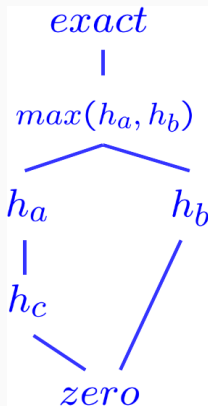
Dizemos que uma heurística h_2 domina a heurística h_1 se $h_2(n) \geq h_1(n)$ para qualquer nó n .

Se tivermos m heurísticas sem relação de dominância entre elas, podemos definir uma heurística h :

$$h(n) = \max\{h_1(n), h_2(n), \dots, h_m(n)\}$$

Criando uma heurística

Podemos pensar no conjunto de heurísticas admissíveis para um problema como o seguinte reticulado:



Sendo a heurística *exacta* o valor exato do custo ao seguir por n e a heurística **zero** uma função $h(n) = 0$.

Exercício

Aplicando a heurística $h(n) = \max\{h1(n), h2(n)\}$ no problema 8 puzzle, obtemos algum ganho?

7	2	4
5	0	6
8	3	1