

Busca la muestra



"BUSCA HEURÍSTICA"

- heurística pode ser definida como o "estudo dos dos métodos e regras de descoberta e invenção"
 - interpretações que nem da origem da palavra
- na busca em espaço de estados, heurísticas são formalizadas como regras para escolher aqueles ramos em um espaço de busca que tem maior probabilidade de levarem a uma solução aceitável do problema
- resoluedores de problemas de IA empregam heurísticas em duas situações:
 - ① problemas que podem não ter uma solução exata por causa das ambiguidades

diferentes na formulação do problema ou nos dados disponíveis. Ex:

- diagnóstico médico : Sintomas podem ter diferentes causas
- visão : cenas visuais são ambíguas (ilusões de ótica).

② Problema possui uma solução exata, mas o custo computacional de encontrá-la é muito alto (ou proibitivo). Ex:

- Xadrez

• Infelizmente, as heurísticas podem falhar.

Uma heurística é apenas uma conjectura informada sobre o próximo passo a ser dado na solução de um problema.

→ frequentemente ela é baseada na experiência e intuição

→ uma heurística pode levar um algoritmo a uma solução sub-ótima, ou até mesmo a não encontrar nenhuma solução.

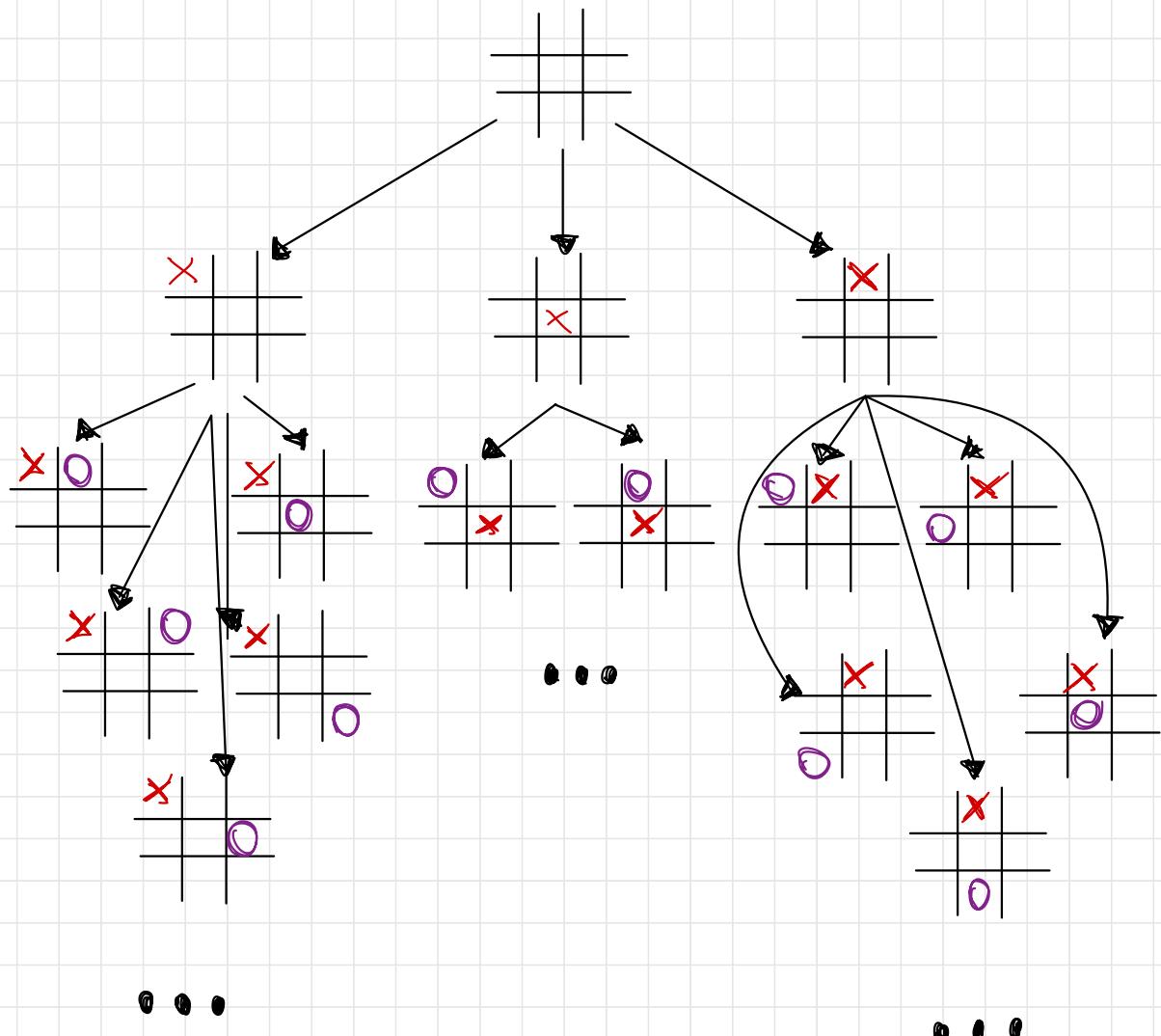
- É útil considerar a busca heurística sob duas perspectivas:
 - * a medida heurística
 - * um algoritmo que usa heurísticas para buscar um espoço de estados

- Alguns exemplos:

Subida de Encosta } Heurísticas
Programação Dinâmica }

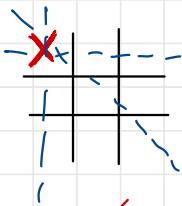
Busca pela Melhor Escolha } Algoritmo

EXEMPLO: 3 primeiros níveis do espaço de estados
do jogo da velha reduzido por simetria

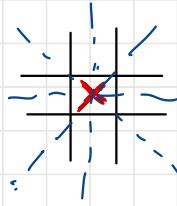


- Usando a representação do espaço por simetria reduz o espaço de estados de $9!$ para $12 \times 7!$
- uma heurística simples pode reduzir ainda mais a busca: podemos nos mover para a configuração na qual X tem mais oportunidades de vitória
- Isto pode ser avaliado na figura 2.

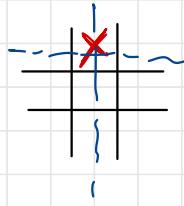
Fig 2: Heurística do "mais vitórias" aplicadas no nível 1 do jogo da velha



três vitórias
para um X no
canto



quatro vitórias
para o X no
centro



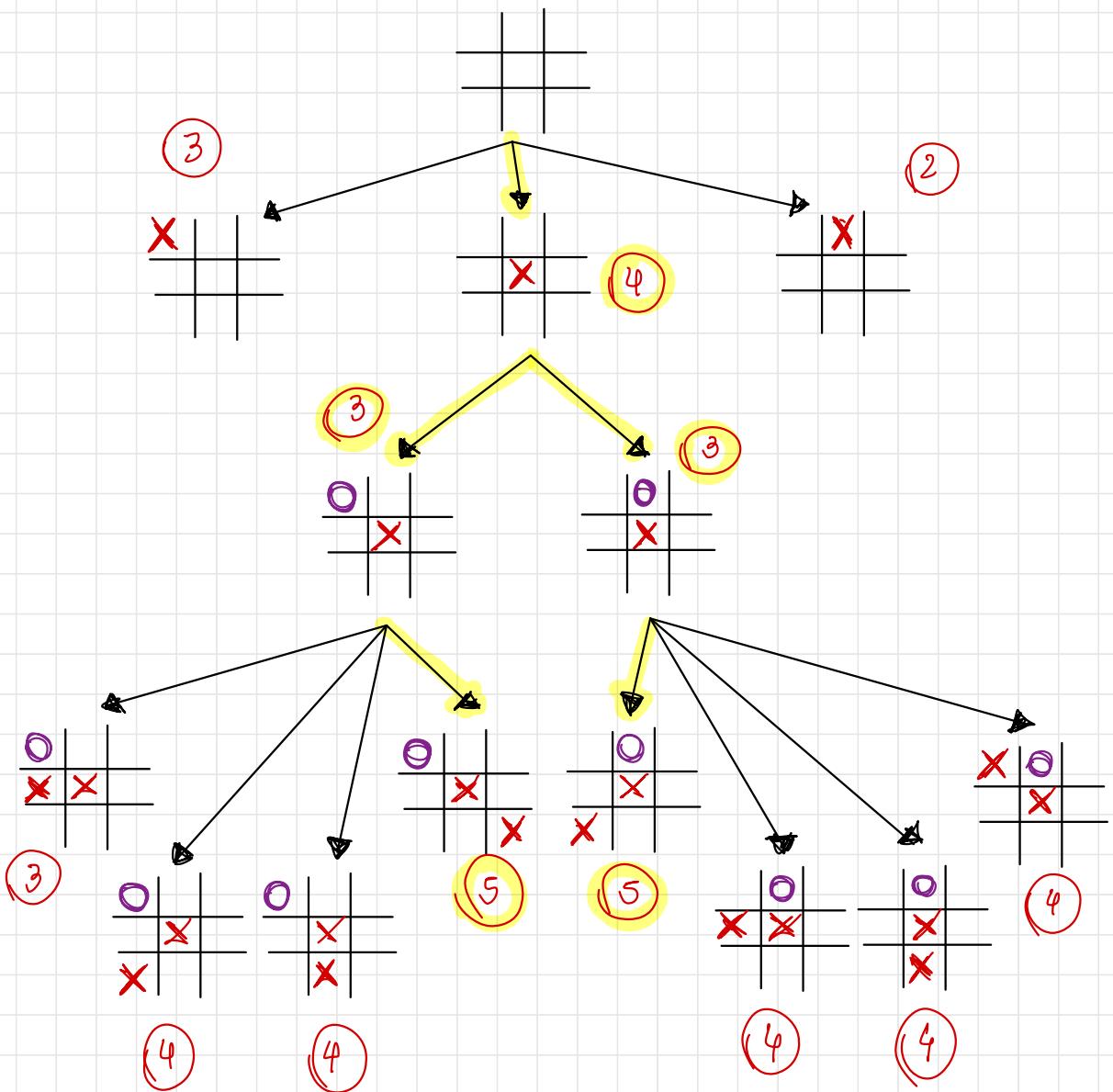
duas vitórias
para um X no
centro de um
lado

- Independentemente de qual movimento seja escolhido, podemos aplicar a heurística ao estado resultante do jogo novamente usando a heurística das "mais vitória" para escolher os movimentos possíveis

- Conforme a busca continua, cada movimento avalia os filhos de um único nó, e uma busca exaustiva não é necessária
- Embora seja difícil calcular o número exato de estados que devem ser examinados, pode-se calcular um limite superior grosseiro, supondo-se um número máximo de 5 movimentos em um jogo com cinco opções por movimento:

$$25 < 9!$$

Fig 3. Espaço de Estados redimensionado teoricamente para o jogo da velha



Exercício 01: Fazer o esboço de busca heurística da Figura 3 até um estado objetivo

Exercício 02: Encontrar e avaliar uma heurística para o problema do quebra-cabeça de 8 peças.

Sugestão $h_1(x)$: Número de peças na posição correta

* 1. Subida de Encosta (Hill Climbing)

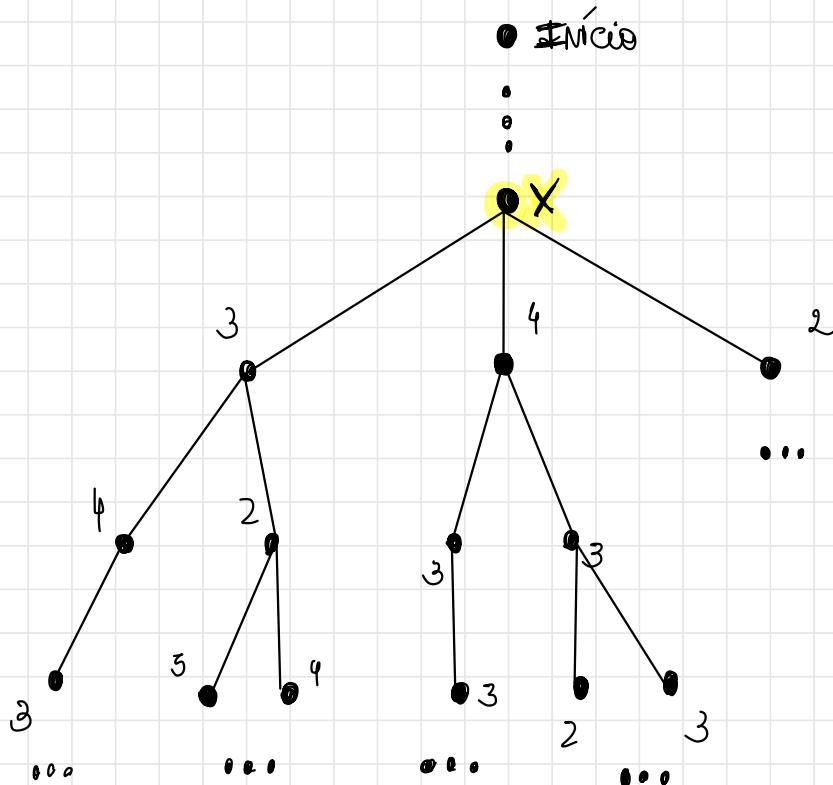
- a maneira mais simples de se implementar a busca heurística é por meio de um algoritmo chamado "subida de encosta"
 - ↳ expande o estado atual da busca e avalia seus filhos
 - ↳ o "melhor" filho é selecionado para uma expansão futura
 - ↳ nem seus irmãos, nem seus genitores são considerados
- analogia: alpinista cego, sempre segue o caminho mais íngreme até não poder avançar mais
- Como o algoritmo não registra o histórico do processo de subida, ele não consegue rever gran de falhas de sua estratégia

- O jogo da velha usando a heurística do caminho com o maior número de vitórias é um exemplo de estratégia de subida de encosta

- um problema da subida de encosta é sua tendência de ficar preso nos máximos locais
 - se ela alcança um estado que tenha uma avaliação melhor que qualquer um de seus filhos, o algoritmo fracassa
 - se esse estado não for um objetivo, mas apenas um máximo local, o algoritmo pode nunca achar a solução
- A figura 4 mostra um exemplo do dilema do máximo local. Suponha que explorando o espaço de busca chegamos ao estado X.

→ as avaliações dos filhos, netos e bisnetos de X demonstraram que a subida de vencosta pode se confundir com a antecipação de vários níveis

Fig 4. O problema do máximo local para a subida de vencosta com antecipações de 3 níveis



- há formas de contornar esse problema, como perturbar aleatoriamente a função de avaliação, mas em geral não há um modo de garantir o desempenho ideal com técnicas de subida de encosta

- * programa de Damas de Samuel (1959)
 - aplicava busca heurística ao jogo de damas
 - implementava algoritmos para o uso ideal da memória limitada, e uma forma simples de aprendizado
 - programa avaliava estados do tabuleiro como a soma ponderada de várias medidas heurísticas diferentes:

$$\sum_i a_i x_i$$

→ o α_i representa características do tabuleiro

Com elas:

- vantagem das peças
- local das peças
- controle do centro do tabuleiro
- etc

→ os coeficientes α_i desses α_i eram pesos ajustados que modelavam a importância desse fator na avaliação geral do tabuleiro

→ o programa antecipava o número desejado de níveis, ou camadas, e avaliava a melhor jogada

- era uma variação do minimax

Exercício: Implemente o algoritmo de Subida de Encosta para o quebra-cabeça de 8 peças.

2	8	3
1	6	4
7		5

estado
inicial

1	2	3
8		4
7	6	5

estado
objetivo

* 2. Algoritmo de Busca pela Melhor Escolha

↳ apesar de suas limitações, os algoritmos como busca com retrocesso, subida de encosta e programação dinâmica podem ser usados eficientemente se suas funções de avaliação forem suficiente informativas para evitar: máximos locais, becos sem saída e anomalias do espaço de busca

↳ em geral, o uso da busca heurística requer um algoritmo mais flexível

- busca pela melhor escolha
- fila de prioridade

↳ igual DFS e BFS, a busca pela melhor escolha usa listas para manter seus estados

- ABERTOS: registrar a fronteira atual da busca

• **Fechados**: registrar os estados já visitados

↳ uma etapa adicional ordena os estados em ABERTOS de acordo com uma estimativa

“heurística” de sua proximidade com um objetivo

↳ cada iteração do laço considera o estado “mais promissor” na lista ABERTOS

ALGORITMO BUSCA PELA MELHOR ESCOLHA

busca_melhor_escolha (Início)

1. $\text{ABERTOS} = [\text{Início}]$ // Inicialização
2. $\text{Fechados} = []$
3. Enquanto $\text{ABERTOS} \neq []$ faça

- 4.
- retire o estado mais à esquerda de **ABERTOS** e chame-o de X

5. Se $X = \text{objetivo}$ então:

- 6.
- retorne o caminho de Inicial até X

7. Senão

- 8.
- gera filhos de X

9. Para cada filho de X faça:

10. Caso:

- 11.
- o filho não está em **ABERTOS** ou **FECHADOS**

12. → atribua ao filho um valor heurístico

13. → acrescente o filho a **ABERTOS**

- 14.
- o filho já está um **ABERTOS**

15. → se o filho foi alcançado por um caminho mais certo

16.

→ então dê ao estado em
Abertos o caminho mais
curto

17.

- o filho já está em **Fechados**
→ se o filho foi alcançado
por um caminho mais cur-
to, então

18.

19.

* retire o estado de
Fechados

20.

* Aumente o filho em
Abertos

21.

fim caso

22.

fim para

23.

fim se/senão

24.

- Coloque X em **Fechados**

25.

- reordene estados em **Abrertos** pelo método heurístico (melhor mais à esquerda)

26.

fim enquanto

27.

retorna **FALHA**

// Abrertos está vazio

28. fim algoritmo

- a cada iteração o algoritmo **busca melhor escolha** retira o primeiro elemento da lista

Abrertos.

→ Se encontrar as condições de objetivo, o algoritmo reforma o caminho de solução que levou ao objetivo

OBS: note que cada estado retém informações do ancestral para determinar se ele tinha sido alcançado anteriormente por um caminho

mais curto e permitir que o algoritmo encontre o caminho de solução

→ se o primeiro elemento em Aceitos não for um objetivo, o algoritmo aplica todas as regras ou operadores de produção que combinam para gerar seus descendentes

→ se um estado filho não estiver em Aceitos ou Fechados, o algoritmo aplica uma avaliação heurística a esse estado, e a lista Aceitos é ordenada. Isto leva os melhores estados para o começo da lista

OBS: Note que este "melhor" pode ser de qualquer nível do espaço de busca. A lista Aceitos é mantida como uma fila de prioridades

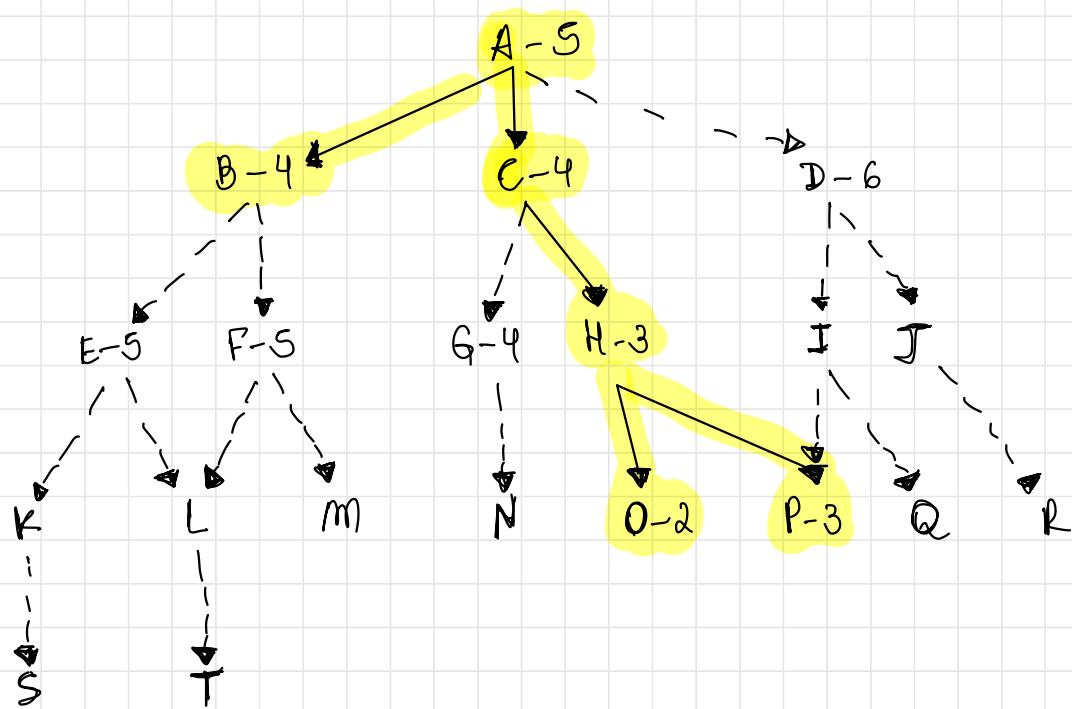
→ se um estado filho já estiver em ~~Abertos~~
ou ~~Fechados~~, o algoritmo verifica para ter
certeza de que o estado registra o mais
curto dos dois caminhos parciais da solução.

Estados duplicados não são refidos.

OBS: atualizando o histórico de caminho dos
nós em ~~Abertos~~ e ~~Fechados~~ quando eles
são redescobertos, o algoritmo achará um
caminho mais curto para um objetivo.

- a Figura 5 mostra um espaço de
estados hipotéticos com avaliações heurísticas
ligadas a alguns de seus estados. Os esta-
dos conectados são gerados pelo processo de
busca. Os estados deslocados foram expandidos
pelo algoritmo

Fig 5. Busca heurística de um espaço de estados hipotético



- Note que o algoritmo não explora todo o espaço de busca. O objetivo do algoritmo é encontrar o estado objetivo examinando o mínimo de estados possível
 - quanto mais “informada” a heurística, menos estados são processados

- Execução do Espaço Hipotético da Figura 5
→ P é o votado objetivo