

SICO7A

SISTEMAS INTELIGENTES 1

Aula 03 F - A*

Prof. Rafael G. **Mantovani**

Roteiro



- 1** Introdução
- 2** Implementando Funções Heurísticas
- 3** A^*
- 4** Exercício
- 5** Referências

Roteiro

- 1** Introdução
- 2** Implementando Funções Heurísticas
- 3** A^*
- 4** Exercício
- 5** Referências

Introdução



Heurísticas simples:

Introdução

Heurísticas simples:

- ... não exploram todas as informações disponíveis em um problema
Exemplo: *posição das peças em um tabuleiro, distância que as peças devem ser movidas, etc ...*
- ... Uma heurística melhor poderia conduzir melhor o processo de busca

Introdução



Introdução



Roteiro

- 1 Introdução
- 2 Implementando Funções Heurísticas
- 3 A*
- 4 Exercício
- 5 Referências

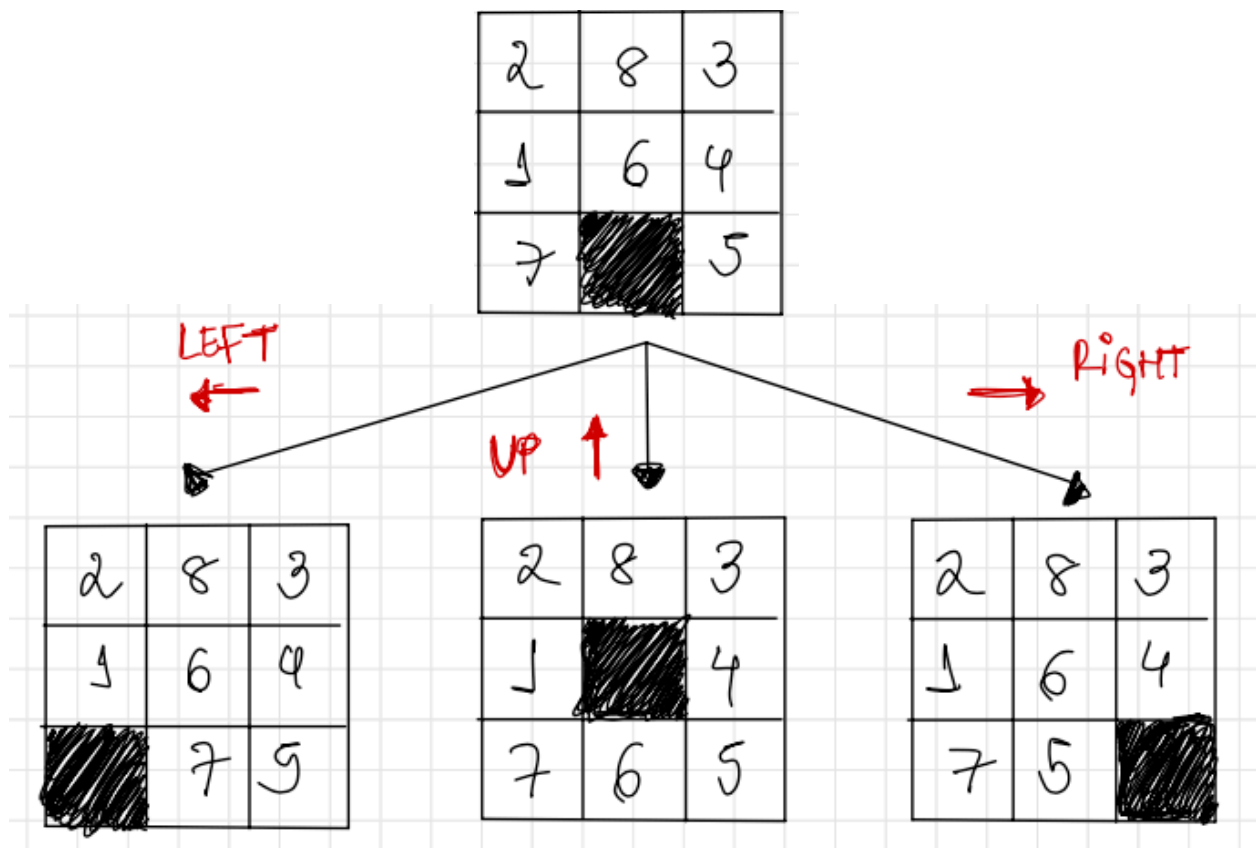
Funções Heurísticas

Fig 1. Estado do quebra-cabeça de 8 peças e movimentos possíveis

2	8	3
1	6	4
7		5

Funções Heurísticas

Fig 1. Estado do quebra-cabeça de 8 peças e movimentos possíveis



Funções Heurísticas

- Três diferentes funções heurísticas
 - **$h1(n)$** : número de peças fora do lugar
 - **$h2(n)$** : soma das distâncias fora do lugar
 - **$h3(n)$** : 2x o número de inversões diretas

Funções Heurísticas

□ $h_2(n)$:

2	8	3
1		4
7	6	5

1	2	3
8		4
7	6	5

objetivo

Funções Heurísticas

□ **$h_3(n)$:**

2	1	3
8		4
7	5	6

1	2	3
8		4
7	6	5

objetivo

Funções Heurísticas

Fig 2. Comparativo de Heurísticas

e1

2	8	3
1	6	4
	7	5

e2

2	8	3
1		4
7	6	5

e3













2	8	3
1	6	4
7	5	

Objetivo

1	2	3
8		4
7	6	5

Funções Heurísticas

Fig 2. Comparativo de Heurísticas

e1	<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td></td><td>7</td><td>5</td></tr></table>	2	8	3	1	6	4		7	5			
2	8	3											
1	6	4											
	7	5											
e2	<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td></td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	2	8	3	1		4	7	6	5			
2	8	3											
1		4											
7	6	5											
e3	<table><tr><td>2</td><td>8</td><td>3</td></tr><tr><td>1</td><td>6</td><td>4</td></tr><tr><td>7</td><td>5</td><td></td></tr></table>	2	8	3	1	6	4	7	5				
2	8	3											
1	6	4											
7	5												
OBJETIVO		h1(n)	h2(n)	h3(n)									
<table><tr><td>1</td><td>2</td><td>3</td></tr><tr><td>8</td><td></td><td>4</td></tr><tr><td>7</td><td>6</td><td>5</td></tr></table>	1	2	3	8		4	7	6	5				
1	2	3											
8		4											
7	6	5											

Funções Heurísticas

Fig 2. Comparativo de Heurísticas

e1	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>1</div><div>6</div><div>4</div> </div> <div> <div></div><div>7</div><div>5</div> </div>	5	6	0
e2	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>1</div><div></div><div>4</div> </div> <div> <div>7</div><div>6</div><div>5</div> </div>	3	4	0
e3	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>1</div><div>6</div><div>4</div> </div> <div> <div>7</div><div>5</div><div></div> </div>	5	6	0
Objetivo	<div> <div>1</div><div>2</div><div>3</div> </div> <div> <div>8</div><div></div><div>4</div> </div> <div> <div>7</div><div>6</div><div>5</div> </div>	$h1(n)$	$h2(n)$	$h3(n)$

Funções Heurísticas

Fig 2. Comparativo de Heurísticas

e1	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>1</div><div>6</div><div>4</div> </div> <div> <div></div><div>7</div><div>5</div> </div>	5	6	0
e2	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>7</div><div>6</div><div>5</div> </div> <div> <div>2</div><div>8</div><div>3</div> </div>	* Qual a melhor heurística?		
e3	<div> <div>2</div><div>8</div><div>3</div> </div> <div> <div>1</div><div>6</div><div>4</div> </div> <div> <div>7</div><div>5</div><div></div> </div>	5	6	0
Objetivo	<div> <div>1</div><div>2</div><div>3</div> </div> <div> <div>8</div><div></div><div>4</div> </div> <div> <div>7</div><div>6</div><div>5</div> </div>	h1(n)	h2(n)	h3(n)

Funções Heurísticas

□ Observações

- a heurística h_2 fornece uma estimativa mais precisa do que a heurística h_1
- a heurística h_3 não consegue distinguir entre os estados, por nenhum deles possui uma situação de inversão direta
- uma quarta heurística h_4 poderia contornar os problemas individuais, combinando h_2 e h_3

Funções Heurísticas

□ Observações

- a heurística h_2 fornece uma estimativa mais precisa do que a heurística h_1

*** Se dois estados tiverem a mesma, ou quase a mesma, avaliação heurística, qual escolher?**

- uma quarta heurística h_4 poderia contornar os problemas individuais, combinando h_2 e h_3

Funções Heurísticas

□ Solução

* distância do caminho atual é mentida por uma contagem da **profundidade do estado**

Funções Heurísticas

□ Solução

* distância do caminho atual é mentida por uma contagem da **profundidade do estado**

- estado inicial tem uma profundidade 0
- contagem é incrementada em 1 para cada nível de busca
- podemos adicionar essa contagem à avaliação heurística de cada estado, para orientar a busca em favor de estados mais superficiais

Funções Heurísticas



Avaliação heurística:

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

Funções Heurísticas

Avaliação heurística:

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



mede o comprimento do
caminho de um estado n
até o estado inicial


Funções Heurísticas

Avaliação heurística:

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



mede o comprimento do
caminho de um estado n
até o estado inicial



é uma estimativa heurística
da distância entre o estado
 n e o um objetivo


Funções Heurísticas

Avaliação heurística:

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



mede o comprimento do
caminho de um estado n
até o estado inicial

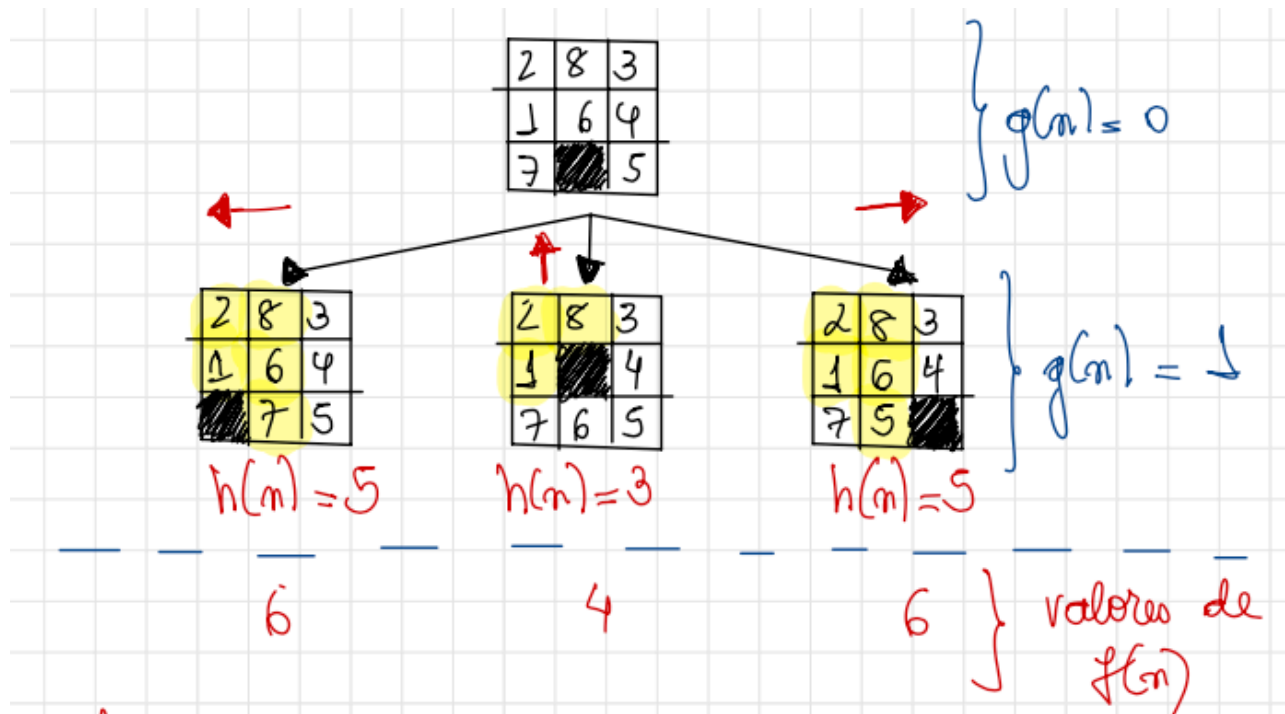


é uma estimativa heurística
da distância entre o estado
 n e o um objetivo

* $h(n)$ pode ser o número de peças fora do lugar
(quebra-cabeça de 8 peças)

Exemplo

Fig 2. Heurística $f(n)$ aplicada no quebra cabeça de 8 peças



onde:

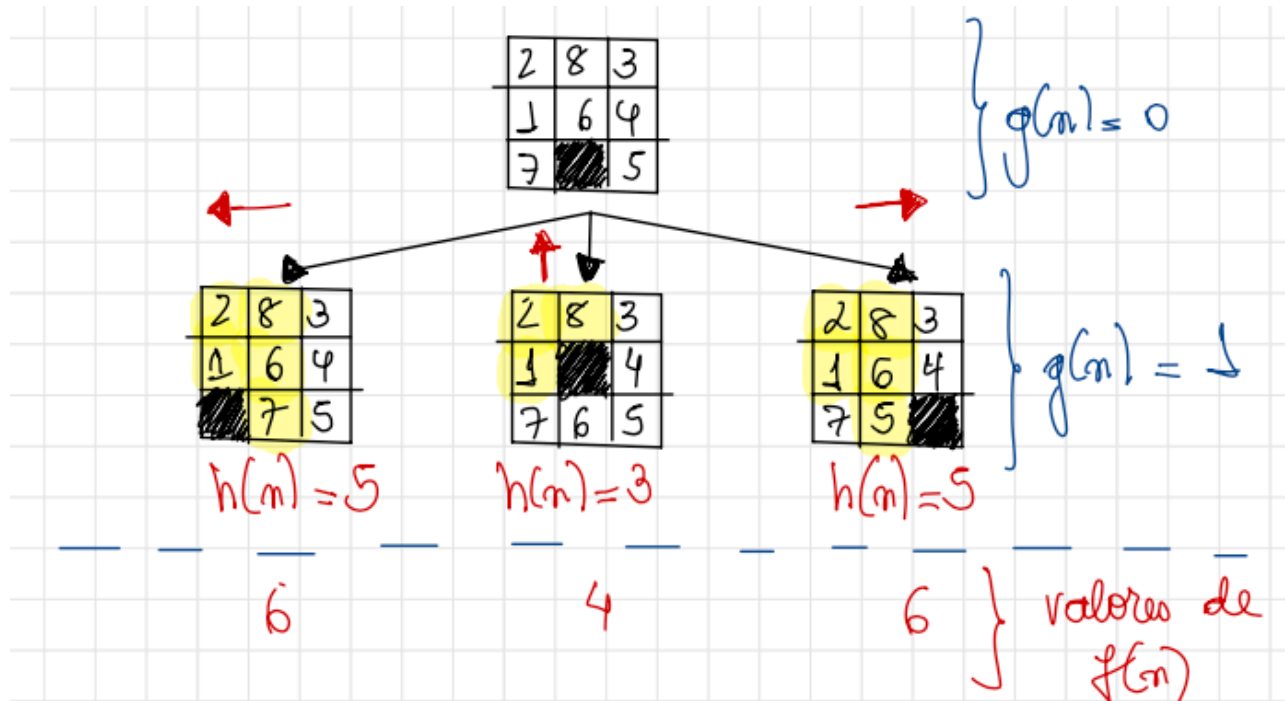
$$f(n) = g(n) + h(n)$$

$g(n)$ = distância real de n até o estado inicial

$h(n)$ = número de peças fora do lugar

Exemplo

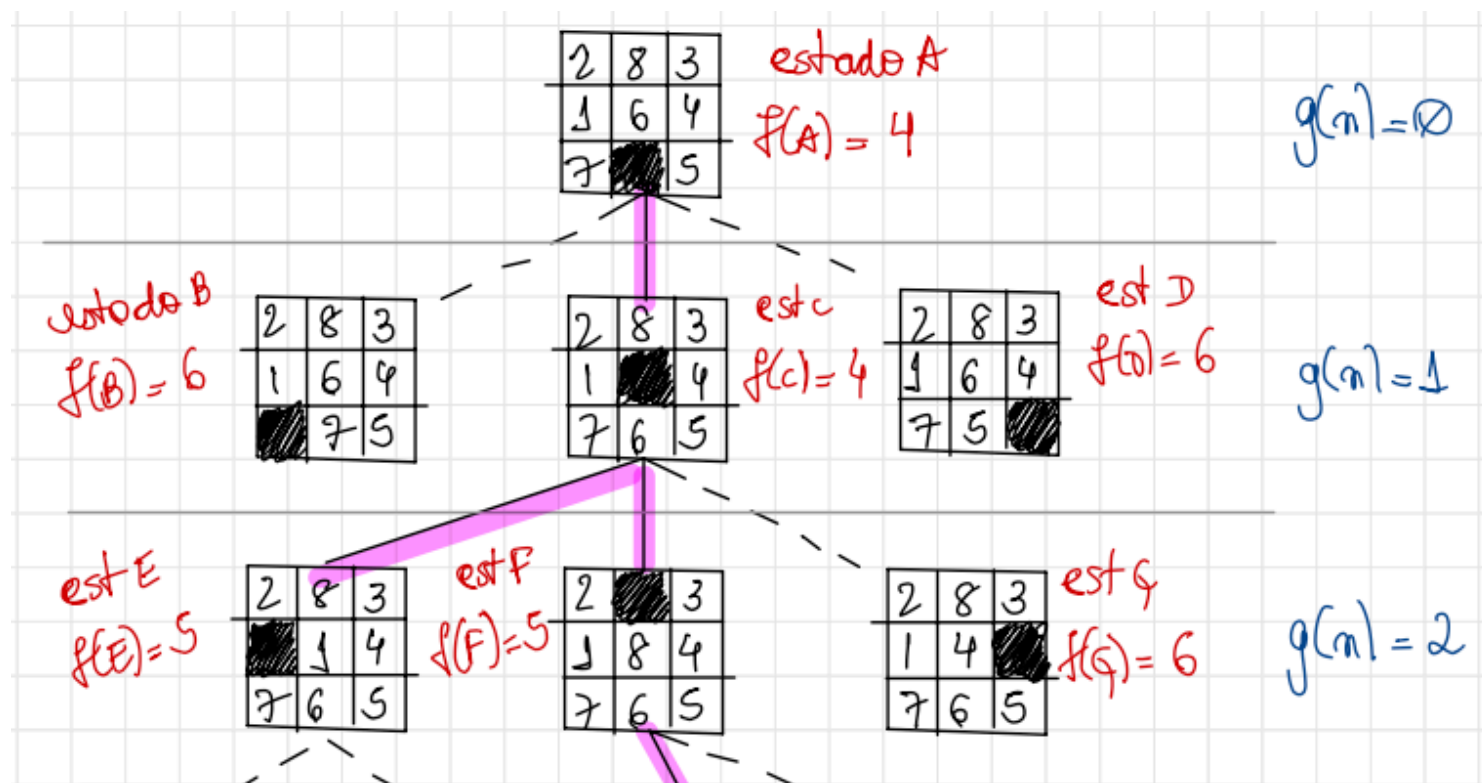
Fig 2. Heurística $f(n)$ aplicada no quebra cabeça de 8 peças



Como continua o espaço de busca?

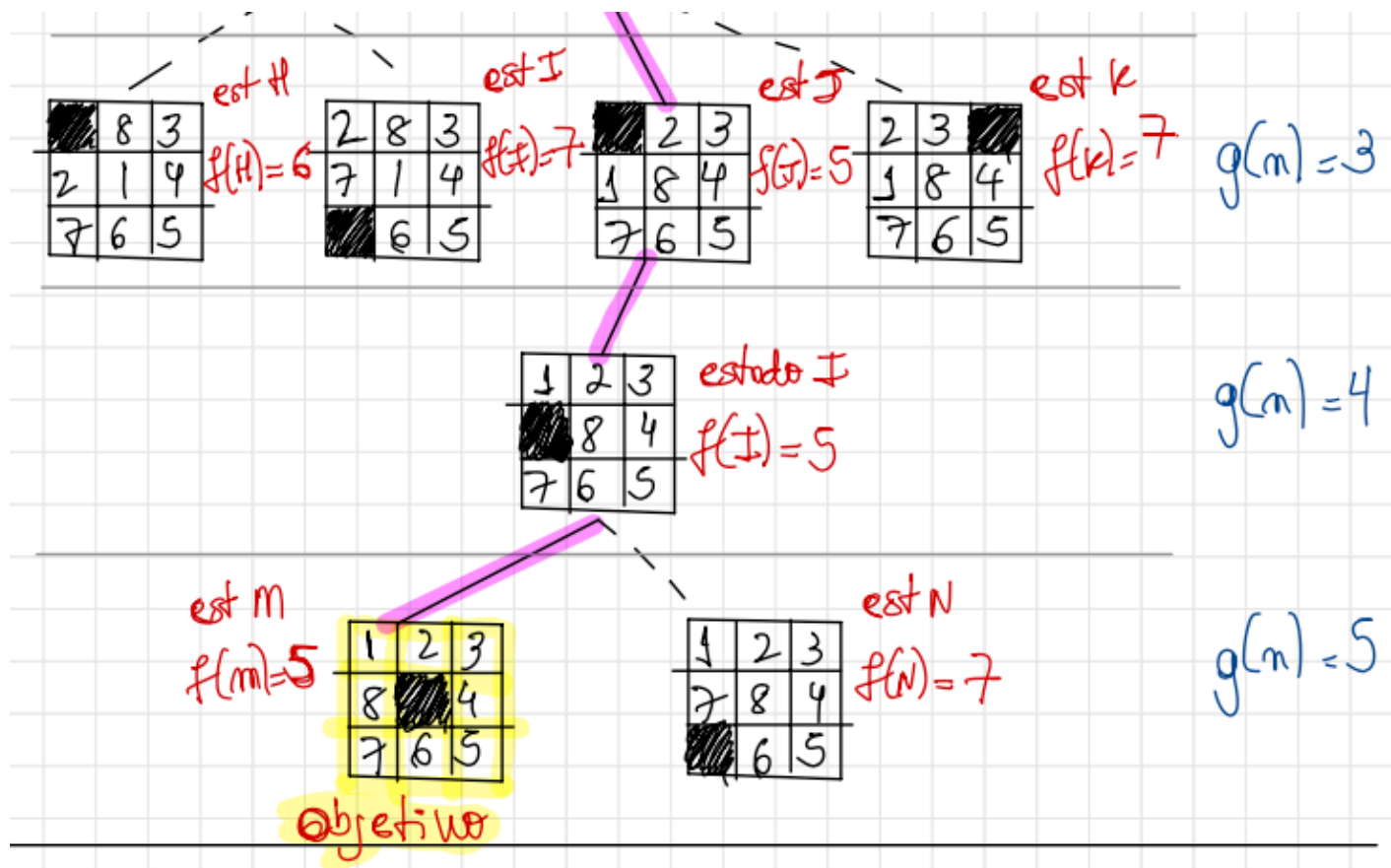
Funções Heurísticas

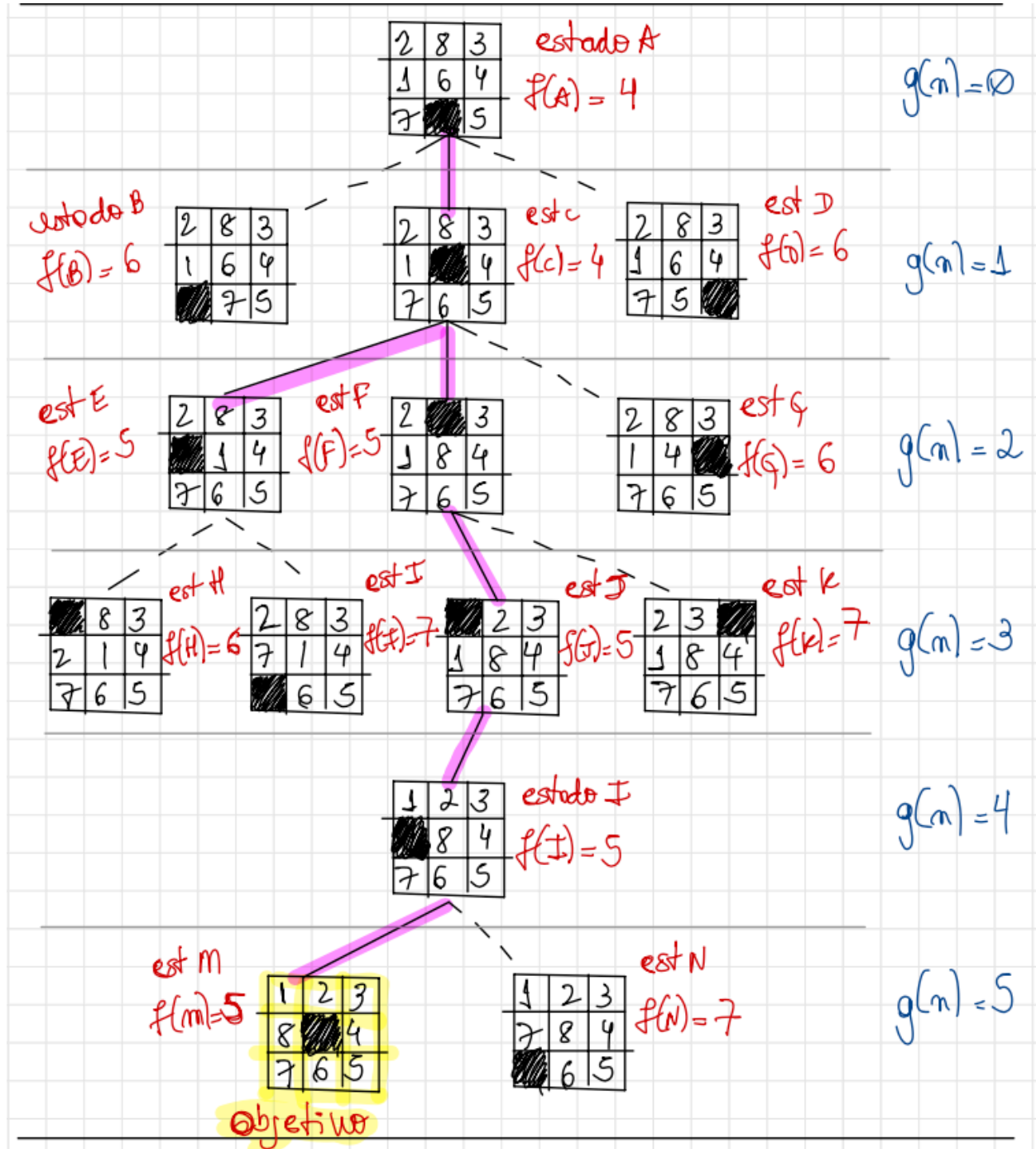
Fig 3. Espaço de estados gerados pela heurística $f(n)$

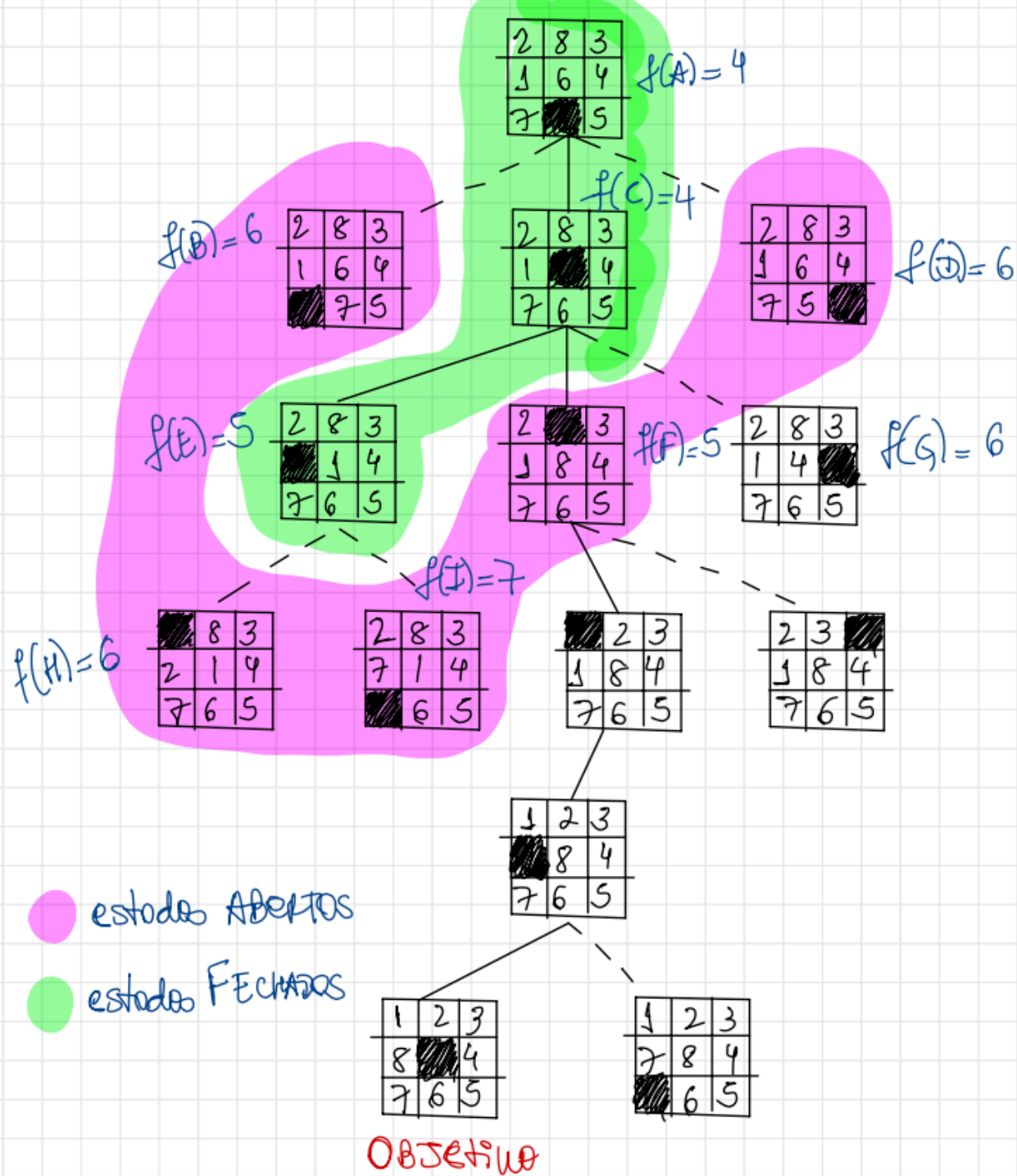


Funções Heurísticas

Fig 3. Espaço de estados gerados pela heurística $f(n)$







Funções Heurísticas

- A **Busca pela Melhor Escolha**, usando a heurística $f(n)$, **pode** garantir a produção do caminho mais curto até um objetivo:

Funções Heurísticas

- A **Busca pela Melhor Escolha**, usando a heurística $f(n)$, **pode** garantir a produção do caminho mais curto até um objetivo:

1

Operações sobre os estados gerais filhos do estado atualmente examinado

Funções Heurísticas

- A **Busca pela Melhor Escolha**, usando a heurística $f(n)$, **pode** garantir a produção do caminho mais curto até um objetivo:

1

Operações sobre os estados gerados filhos do estado atualmente examinado

2

Cada estado novo é verificado para ver se ocorreu antes (está em **ABERTOS** ou **FECHADOS**), impedindo laços

Funções Heurísticas

- A **Busca pela Melhor Escolha**, usando a heurística $f(n)$, **pode** garantir a produção do caminho mais curto até um objetivo:
 - 1** Operações sobre os estados gerados filhos do estado atualmente examinado
 - 2** Cada estado novo é verificado para ver se ocorreu antes (está em **ABERTOS** ou **FECHADOS**), impedindo laços
 - 3** Cada estado n recebe um valor $f(n)$ igual a soma de sua profundidade no espaço de busca $g(n)$, e uma estimativa heurística de distância até um objetivo, $h(n)$

Funções Heurísticas

- A **Busca pela Melhor Escolha**, usando a heurística $f(n)$, **pode** garantir a produção do caminho mais curto até um objetivo:
 - 1** Operações sobre os estados gerados filhos do estado atualmente examinado
 - 2** Cada estado novo é verificado para ver se ocorreu antes (está em **ABERTOS** ou **FECHADOS**), impedindo laços
 - 3** Cada estado n recebe um valor $f(n)$ igual a soma de sua profundidade no espaço de busca $g(n)$, e uma estimativa heurística de distância até um objetivo, $h(n)$

$h(n)$: guia a busca a estados promissores

$g(n)$: impede que a busca persista em um caminho infrutífero

Funções Heurísticas

4

Estados em **ABERTOS** são ordenados por seus valores de $f(n)$

Funções Heurísticas

4

Estados em **ABERTOS** são ordenados por seus valores de $f(n)$

5

A eficiência do algoritmo pode ser melhorada se **ABERTOS** e **FECHADOS** sejam implementadas como **HEAPs**

Funções Heurísticas

- 4 Estados em **ABERTOS** são ordenados por seus valores de $f(n)$
- 5 A eficiência do algoritmo pode ser melhorada se **ABERTOS** e **FECHADOS** sejam implementadas como **HEAPs**

* a **Busca pela Melhor Escolha** é um algoritmo genérico para pesquisa, de modo heurístico, qualquer grafo de estados

Roteiro

- 1 Introdução
- 2 Implementando Funções Heurísticas
- 3 A^*
- 4 Exercício
- 5 Referências



Podemos avaliar o comportamento de heurísticas ao longo de várias dimensões:

- Encontrar uma solução boa
- Encontrar o menor caminho
- Encontrar a solução menos custosa
- etc ...



Podemos avaliar o comportamento de heurísticas ao longo de várias dimensões:

■ Encontrar uma solução boa

* **Heurísticas** que encontram o caminho mais curto até um objetivo, sempre que existir, são chamadas de **admissíveis**

■ etc ...



Admissibilidade

- Um algoritmo de busca é **admissível** se ele encontrar, um caminho mínimo até uma solução, sempre que tal solução existir

Admissibilidade

- Um algoritmo de busca é **admissível** se ele encontrar, um caminho mínimo até uma solução, sempre que tal solução existir

A busca é amplitude é uma estratégia de busca admissível

Admissibilidade

- Um algoritmo de busca é **admissível** se ele encontrar, um caminho mínimo até uma solução, sempre que tal solução existir
- Temos uma classe de estratégias de busca admissíveis usando a função de avaliação f^* :

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

Admissibilidade

- Um algoritmo de busca é **admissível** se ele encontrar, um caminho mínimo até uma solução, sempre que tal solução existir
- Temos uma classe de estratégias de busca admissíveis usando a função de avaliação f^* :

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

$g^*(n)$: custo do caminho **mais curto** do nó inicial até estado n
 $h^*(n)$: retorna o **custo real** do menor caminho de n até o **objetivo**

Admissibilidade

- A **Busca Pela Melhor Escolha** com uma função de avaliação f^* , gera uma estratégia de busca **admissível**

Admissibilidade

- A **Busca Pela Melhor Escolha** com uma função de avaliação f^* , gera uma estratégia de busca **admissível**

PROBLEMA:
$$f^*(n) = g^*(n) + h^*(n)$$

Admissibilidade

- A **Busca Pela Melhor Escolha** com uma função de avaliação f^* , gera uma estratégia de busca **admissível**

PROBLEMA:
$$f^*(n) = g^*(n) + h^*(n)$$

Oráculos como f^* **não** existem para a maior dos problemas reais

A*

SOLUÇÃO:

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

A*

SOLUÇÃO:

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

■ $g(n)$ = custo do caminho atual até n é uma estimativa razoável de g^*

A*

SOLUÇÃO:

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

■ $g(n)$ = custo do caminho atual até n é uma estimativa razoável de g^*

■ $h(n)$ = estimativa de custo mínimo até um objetivo

Se $h(n) \leq h^*(n)$, o algoritmo que usa f é chamado de A*

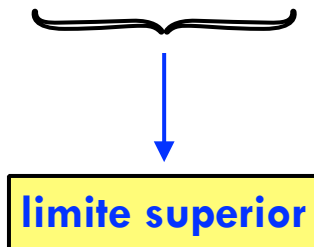
A*

SOLUÇÃO:

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

- $g(n)$ = custo do caminho atual até n é uma estimativa razoável de g^*
- $h(n)$ = estimativa de custo mínimo até um objetivo

Se $h(n) \leq h^*(n)$, o algoritmo que usa f é chamado de A*



RESUMINDO:

- Considerando a função $f(n)$
- Se essa função for usada com o algoritmo **Busca Pela Melhor Escolha**, o resultado é o chamado **algoritmo A**
- Se o algoritmo A for usado com uma função de avaliação $h(n)$, cujos valores são menores que o custo do caminho mínimo de n para um objetivo, ou igual a ele ...
- ... esse algoritmo de busca será chamado de **A* (A ESTRELA)**

A*

RESUMINDO:

- Considerando a função $f(n)$

$h1(n)$, $h2(n)$ e $h3(n)$...
... são **heurísticas admissíveis** para o quebra-cabeça de 8 peças

- ... esse algoritmo de busca será chamado de **A*** (A ESTRELA)

Pseudocódigo: A*

A* (Inicial):

```
1.  ABERTOS = [Inicial] // Inicialização
2.  FECHADOS = [ ]
3.  Enquanto ABERTOS != [ ] faça:
4.      Remova o estado mais à esquerda de ABERTOS, chame-o de X
5.      Se X for um objetivo, então retorne o Caminho de Inicial até X
6.      Senão
7.          Gere filhos de X
8.          Para cada filho de X faça:
9.              Caso:
10.                 → o filho não está em ABERTOS ou FECHADOS:
11.                 atribua ao filho um valor heurístico usando  $f(n) = g(n) + h(n)$ 
12.                 acrescente o filho a ABERTOS
```


Pseudocódigo: A*

A* (Inicial):

13. → o filho já está em **ABERTOS**:
14. **Se** o filho foi alcançado por um caminho mais curto, então:
15. * de ao estado em **ABERTOS** o caminho mais curto
16. → o filho já está em **FECHADOS**:
17. **Se** o filho foi alcançado por um caminho mais curto, então:
18. * retire o estado de **FECHADOS**
19. * adicione o filho em **ABERTOS**
20.
21. **fim caso**
22. **fim para**
23. **fim se-senão**

Pseudocódigo: A*

A* (Inicial):

```
24. | Coloque X em FECHADOS
25. | Reordene os estados em ABERTOS de acordo com o método heurístico
    | (melhor mais à esquerda) // Fila de Prioridade
26. | fim enquanto
27. | retorna FALHA // ABERTOS está vazio
28. fim algoritmo
```

Roteiro

- 1 Introdução
- 2 Implementando Funções Heurísticas
- 3 A^*
- 4 Exercício
- 5 Referências

Exercícios

1) Implemente o algoritmo A^* 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

Exercícios

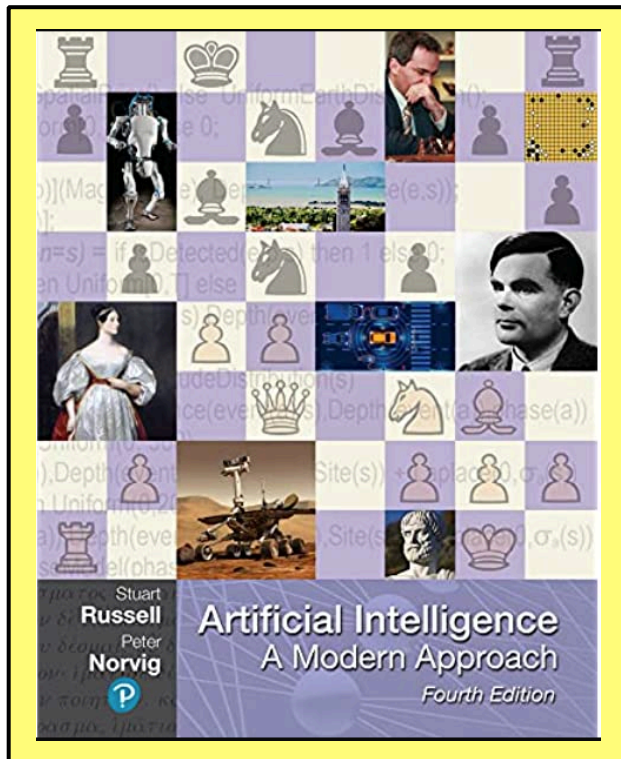


2) Teste seu algoritmo de A^* para outros estados iniciais

Roteiro

- 1 Introdução
- 2 Implementando Funções Heurísticas
- 3 A^*
- 4 Exercício
- 5 Referências

Referências sugeridas



[Russel & Norvig, 2021]



[Luger, 2013]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br