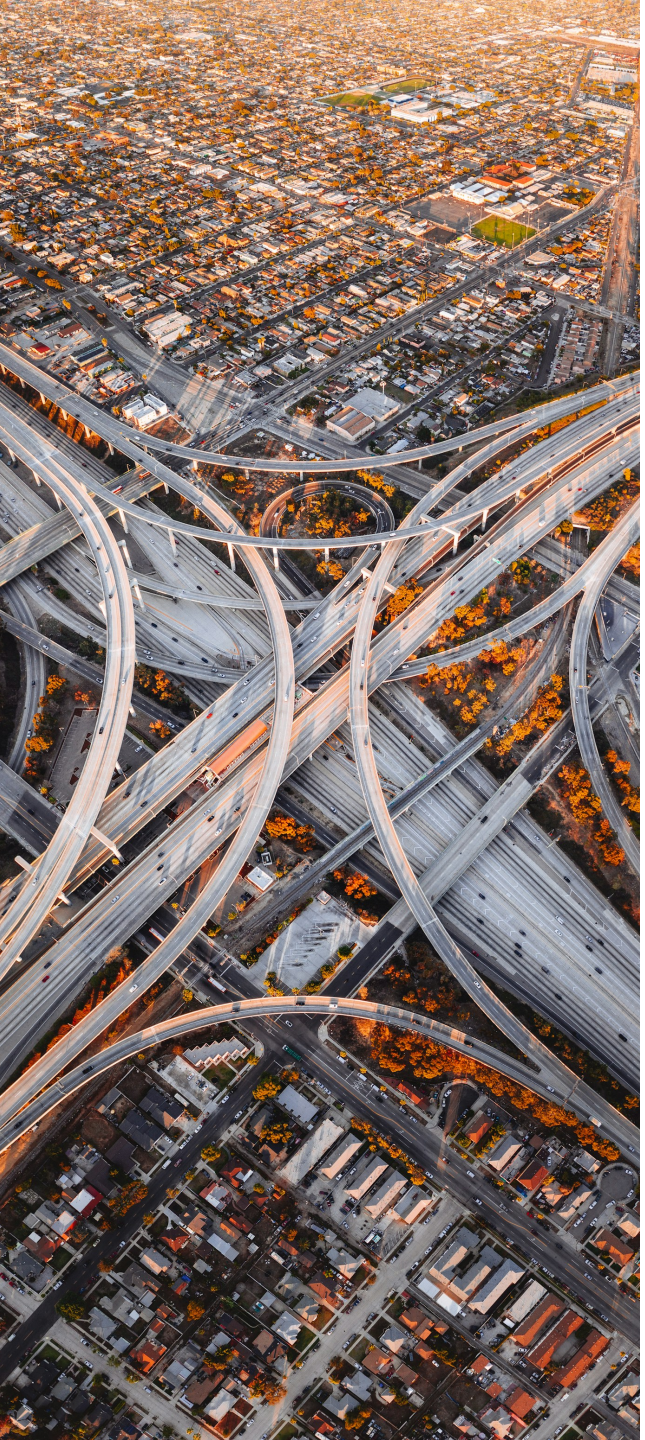


Busca Informada

Inteligência Artificial

Pontifícia Universidade Católica de Campinas

Prof. Dr. Denis M. L. Martins

Objetivos de Aprendizado

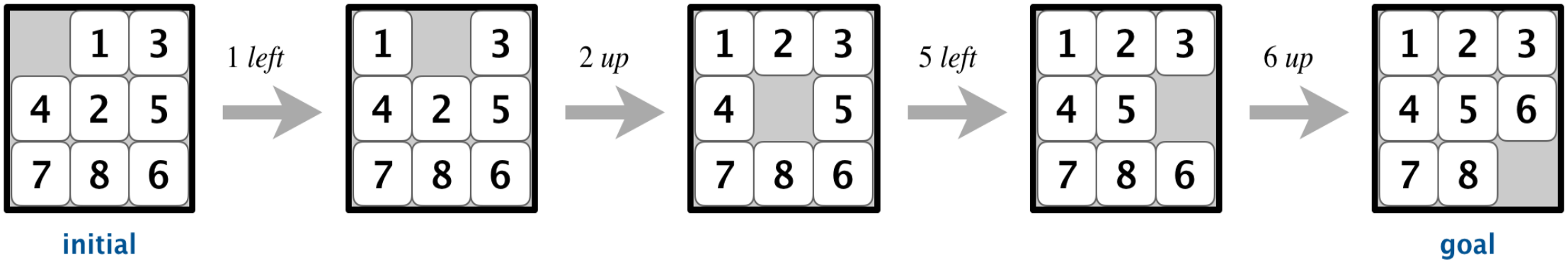
- Entender os **fundamentos teóricos e práticos** das técnicas de busca informada.
- **Diferenciar** entre algoritmos de busca não informada e os de busca informada, reconhecendo suas garantias de completude e optimalidade.
- **Projetar e avaliar funções heurísticas** para problemas clássicos (labirinto, 8-puzzle, planejamento de rotas) e analisar seu impacto no desempenho dos algoritmos.
- Considerar **aplicações de busca informada a cenários do mundo real**, justificando escolhas heurísticas e discutindo limitações práticas (custo computacional, espaço, etc.).

Panorama Geral

- Busca não informada: Explora o espaço de estados sem usar conhecimento adicional (ex.: BFS, DFS).
- Busca informada (heurística)
  - Usa conhecimento do problema para encontrar soluções de forma mais eficiente.
  - **Heurística**: função de avaliação que estima o custo de alcançar o objetivo.
  - Objetivo: reduzir o número de nós expandidos, mantendo ou aproximando-se da otimalidade.
- Metáfora – Navegar sem bússola vs. Navegar com bússola.
  - A heurística é a bússola que aponta na direção do objetivo.

Exemplo: Problema do 8-puzzle

- Quebra-cabeça de lógica composto por nove peças quadradas, numeradas de 1 a 8, dispostas em um tabuleiro (3×3). Uma das posições permanece vazia (na figura abaixo, □ cinza). O objetivo do jogo é reordenar as peças até que o tabuleiro esteja na configuração desejada (na figura, com as peças em ordem crescente).



Exemplo de jogadas no 8-Puzzle. Fonte da imagem: https://8-puzzle.readthedocs.io/en/latest/.

Como medir se estamos **chegando no objetivo**
no contexto do 8-puzzle?

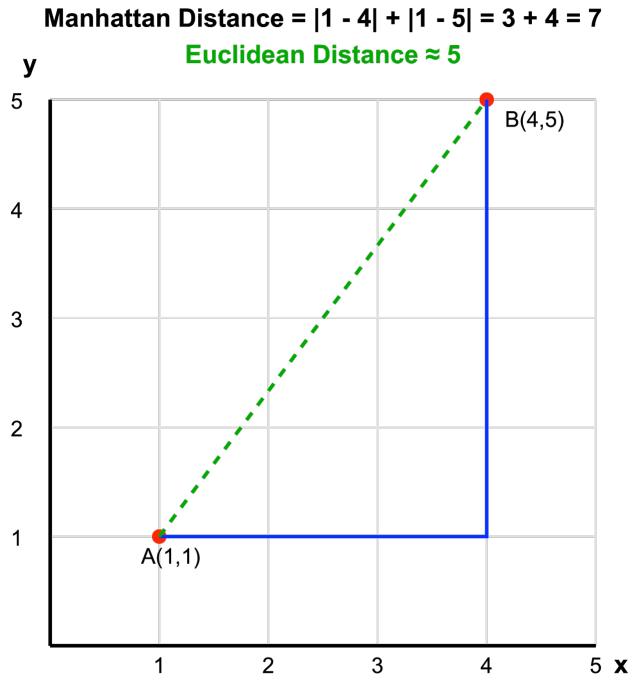
Como medir se estamos **chegando no objetivo** no contexto do 8-puzzle?

Distância de Manhattan

Número de peças fora do lugar

Heurística: Distância de Manhattan

Distância de Manhattam (em vermelho) e distância Euclideana (em azul).
Fonte: Machine Learning with Swift.



Distância de Manhattam (em azul) e distância Euclideana (em verde).
Fonte: DataCamp.

Heurística: Distância de Manhattan

A **distância de Manhattan** mede o número total de movimentos horizontais e verticais necessários para levar cada peça de sua posição atual até a posição correta.

$$h(n) = \sum_{i=1}^{8} \left(|x_i - x_i^*| + |y_i - y_i^*| \right)$$

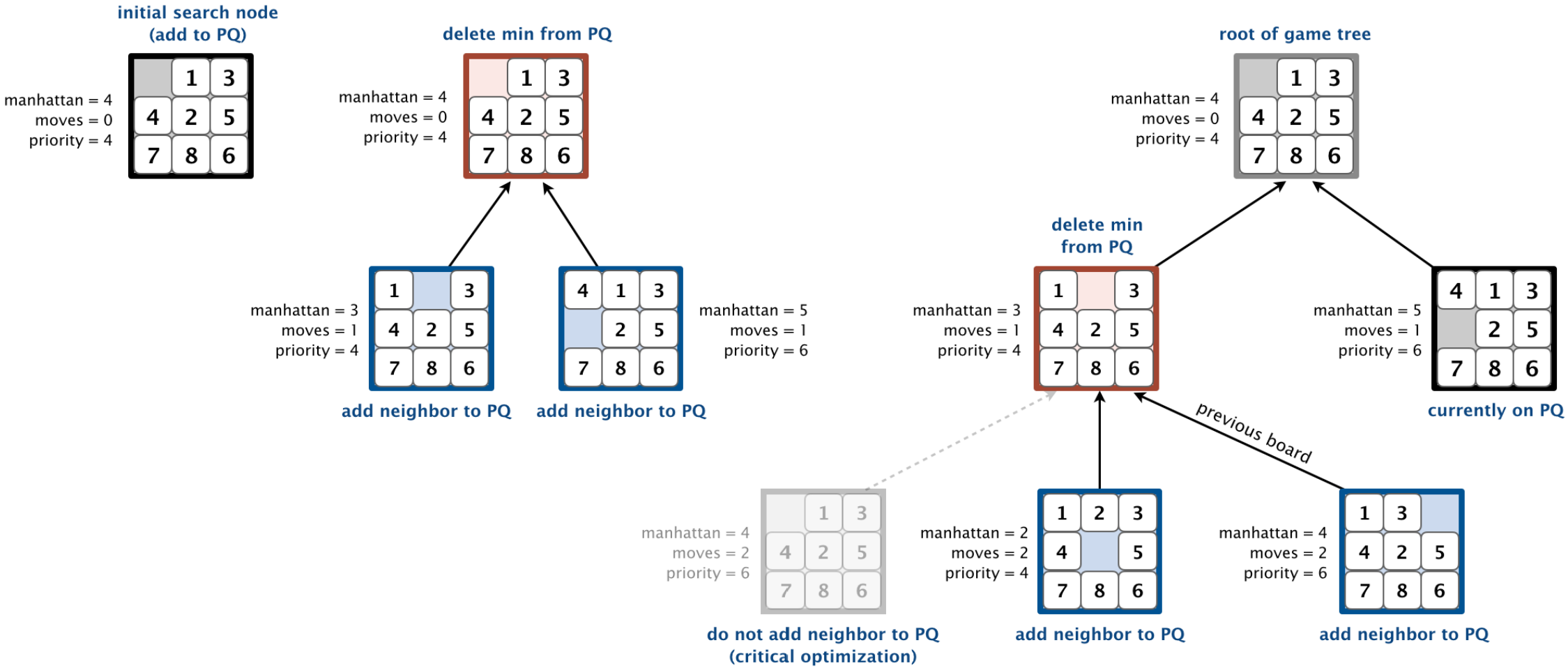
onde: (x_i, y_i) é a posição **atual** da peça i e (x_i^*, y_i^*) = posição **correta** da peça i.

Estado inicial:

```
2 8 3
1 6 4
7   5
```

Objetivo:

```
1 2 3
8   4
7 6 5
```



Fonte da imagem: https://8-puzzle.readthedocs.io/en/latest/

Heurística: Distância de Manhattan

- Admissível?
  - Sim: nunca superestima o custo real.
- Fácil de implementar?
  - Sim: simples e eficiente

Uma heurística é **admissível** se

nunca superestima o custo real até o objetivo.

A Distância de Manhattan é considerada uma heurística **mais eficaz** que apenas contar peças fora do lugar?

Definição Formal de Heurística

Seja $G = (V, E)$ um grafo de estado.

- Função heurística: $h : V \to \mathbb{R}^+$, onde $h(n)$ é a estimativa do custo mínimo restante para alcançar o objetivo a partir de n.
- Admissibilidade: $\forall n,\ h(n) \leq h^*(n)$
  - Não superestima o custo real h^*.
  - Heurísticas admissíveis são otimistas por natureza porque imaginam que o custo de resolver o problema seja **menor** que realmente é.

Exemplo: Distância entre Cidades da Romênia

Considere encontrar o melhor caminho entre as cidades de Arad (estado inicial) e Bucharest (estado objetivo).

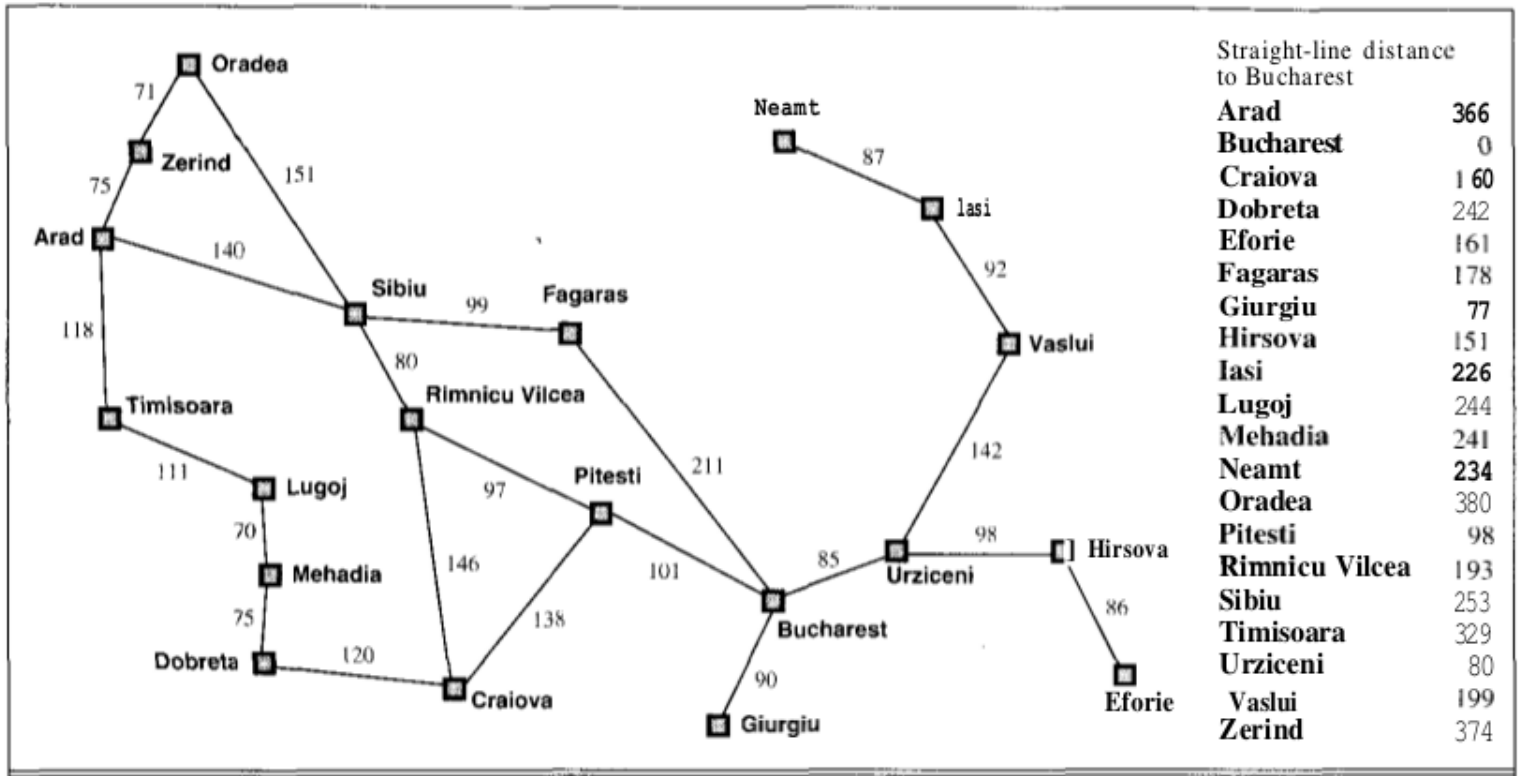


Figure 4.2 Map of Romania with road distances in km, and straight-line distances to Bucharest.

Qual **heurística** você usaria para encontrar um bom (ou o melhor) caminho entre Arad e Bucharest?

Exemplo: Distância entre Cidades da Romênia (cont.)

A tabela à direita da imagem mostra distância em linha reta estimada de cada cidade presente no mapa até à capital do país, Bucharest.

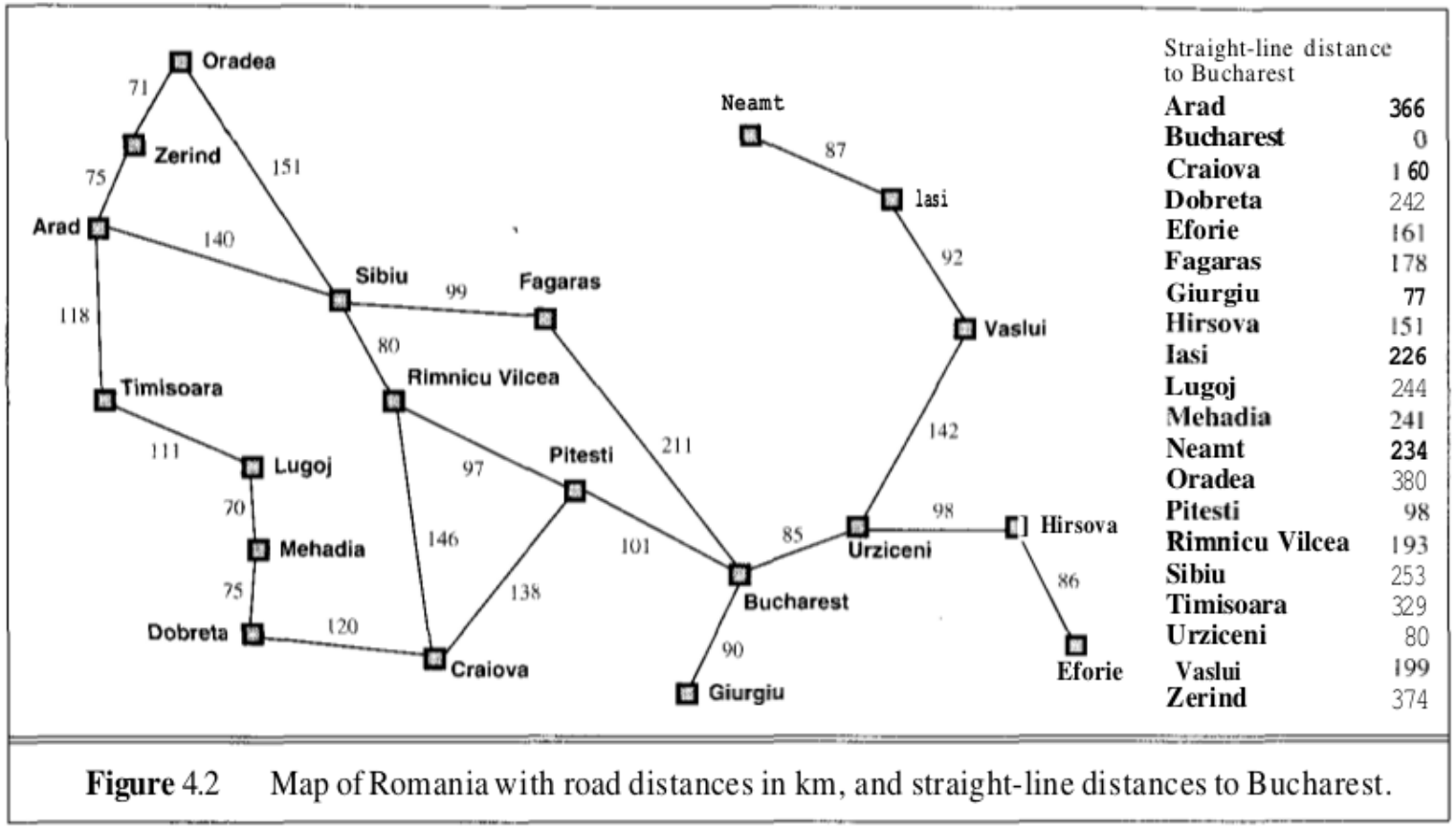


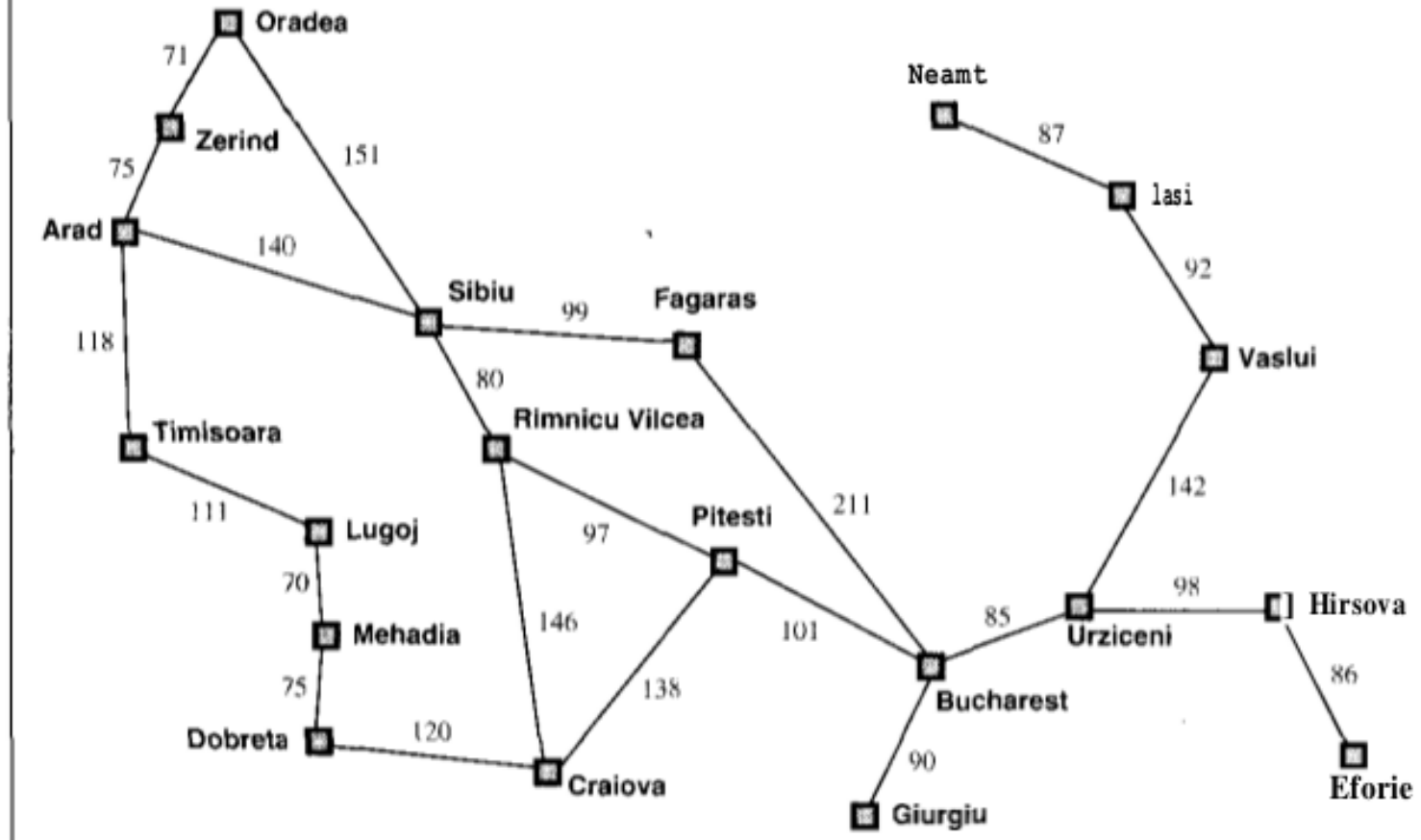
Figure 4.2 Map of Romania with road distances in km, and straight-line distances to Bucharest.

Greedy Best-First Search (Busca Gulosa)

A cada passo escolhe a ação que parece mais promissora no presente.

- Baseia-se em uma função heurística $h(v)$: estimativa do custo restante até o objetivo.
- Não considera custos acumulados, apenas a decisão local.
- **Expande o nó com menor heurística**. A decisão tomada nunca é revista.
- Estrutura de dados: Fila de prioridade (min-heap) Q ordenada pelo valor $h(n)$.
- Completeness: **não** garante solução se houver ciclos e heurística inválida.
- Optimality: **não** garantida (pode escolher caminhos sub-óptimos).
- Aplicações em Tempo Real: Planejamento de rotas de robôs, jogos (IA de NPCs) onde respostas rápidas são importantes.

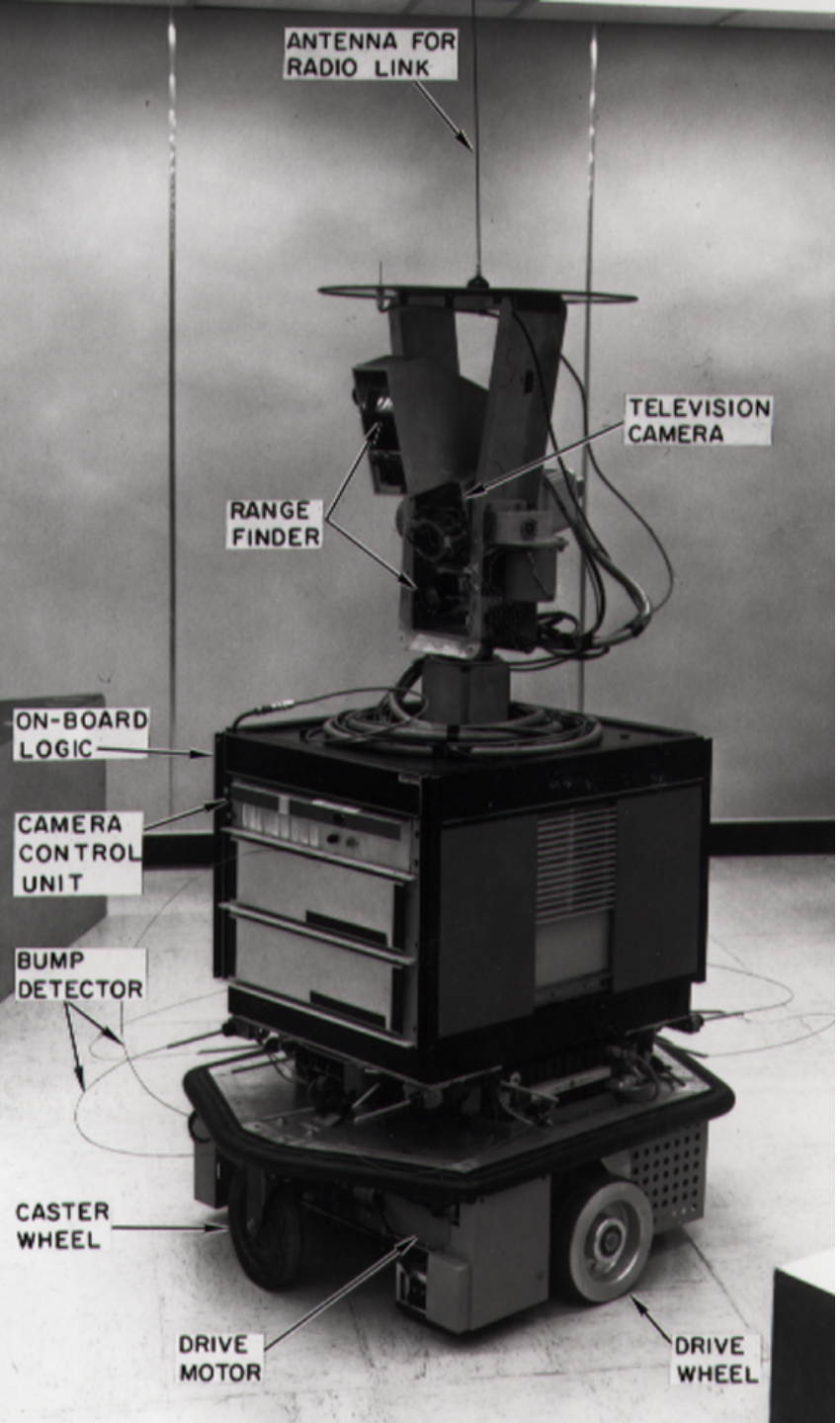
Greedy Best-First: Voltamos ao exemplo anterior. Vamos encontrar o caminho mais curto: (A) de Arad a Bucharest. (B) de Iasi para Fagaras.



Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 4.2 Map of Romania with road distances in km, and straight-line distances to Bucharest.



A* (A-estrela)

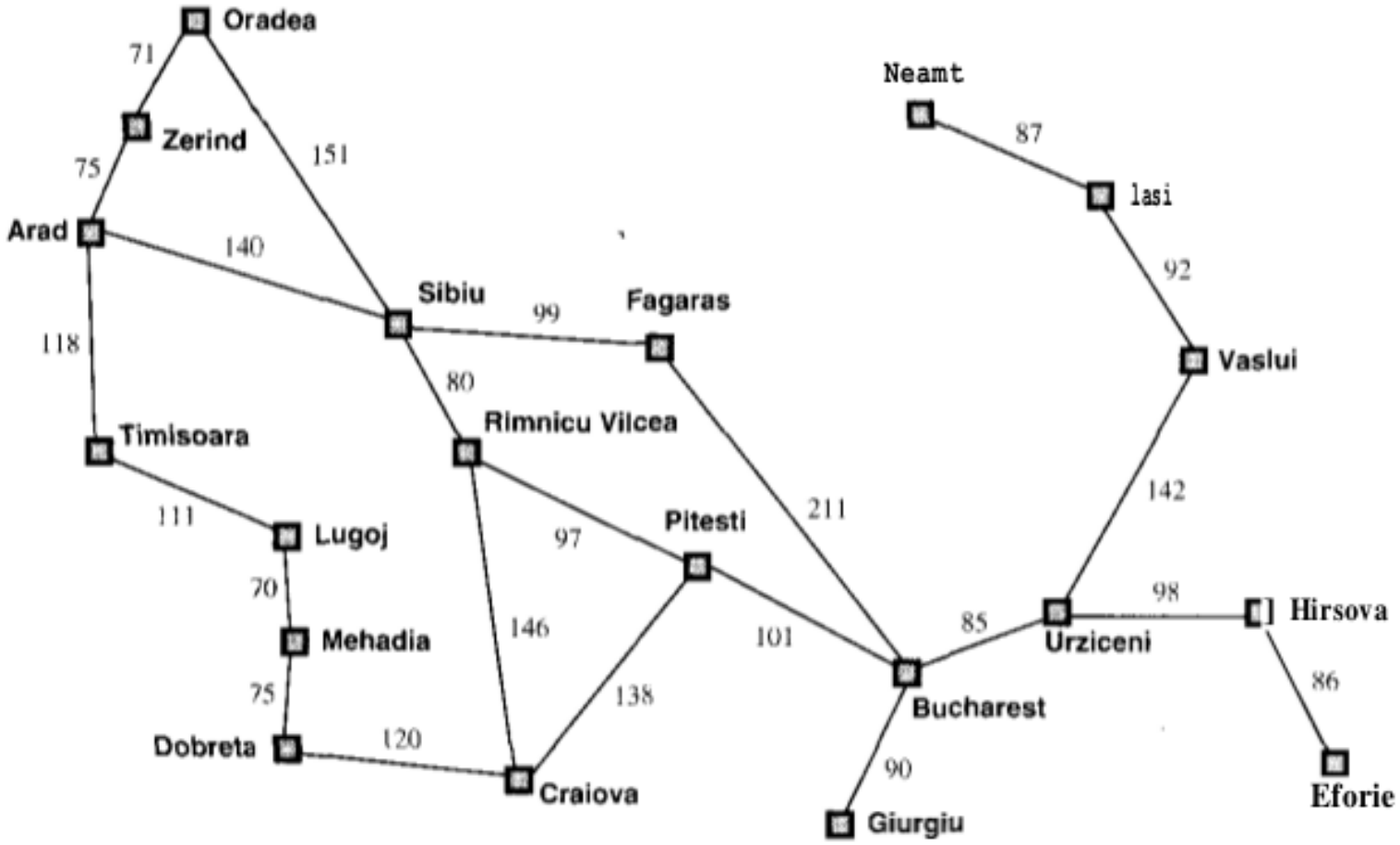
- Função: $f(n) = g(n) + h(n)$
  - g(n): custo acumulado do caminho já percorrido até n.
  - h(n): estimativa heurística do custo de n até o objetivo.
- Estrutura de dados: Fila de prioridade ordenada por $f(n)$.
- Se h for admissível e consistente, A* encontra um caminho ótimo.
  - Se custos forem positivos.
  - Se o fator de ramificação b é finito.
- Consistência (ou Monotonicidade):
  $\forall (n, n'),\ h(n) \leq c(n, n') + h(n')$
  - Observação – A consistência implica admissibilidade.

A* Algoritmo

```
1.  Inserir nó inicial na fila aberta (open).
2.  Enquanto open não vazio:
3.      Selecionar n = node com menor f(n).
4.      Se n é objetivo: reconstruir caminho e terminar.
5.      Expandir n → gerar filhos {n'}.
6.      Para cada filho n':
7.          g' = g(n) + custo(n, n')
8.          h' = heurística(n')
9.          f' = g' + h'
10.         Se n' já em closed com menor f: ignorar
11.         Caso contrário, inserir/atualizar open.
12.     Inserir n em closed.
```

- Open: fila de prioridade.
- Closed: conjunto de nós já examinados.

A*: Voltamos (de novo!?) ao exemplo anterior. Vamos encontrar o caminho mais curto de Arad a Bucharest.



Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Figure 4.2 Map of Romania with road distances in km, and straight-line distances to Bucharest.

A* - Aplicações e Complexidade

Domínio	Uso de A*
Robótica móvel	Planejamento de trajeto em malha 2-D/3-D.
Jogos	Pathfinding em mapas com obstáculos (ex.: RTS, RPG).

Complexidade:

- Tempo: $O(b^d)$ em pior caso, mas com heurística boa pode ser quase linear.
- Espaço: $O(b^d)$ armazena em memória todos os nós gerados.

Exercício: Planejamento de Rota em Grade 2-D com Obstáculos

A* para encontrar o caminho mínimo entre dois pontos em uma grade (matriz) 5×5, onde alguns quadrados são obstáculos.

- `S` - Estado inicial (posição $(0, 0)$).
- `G` - Meta (posição $(3, 4)$).
- `#` - Obstáculos (não podem ser atravessados).
- `.` - Células livres.
- Movimento permitido apenas nas 4 direções cardinais (custo = 1 por passo).
- Qual heurística utilizar?

	0	1	2	3	4
0	**S**	.	.	#	.
1	.	#	.	#	.
2	.	#	.	.	.
3	.	.	.	#	**G**
4	.	#	.	.	.

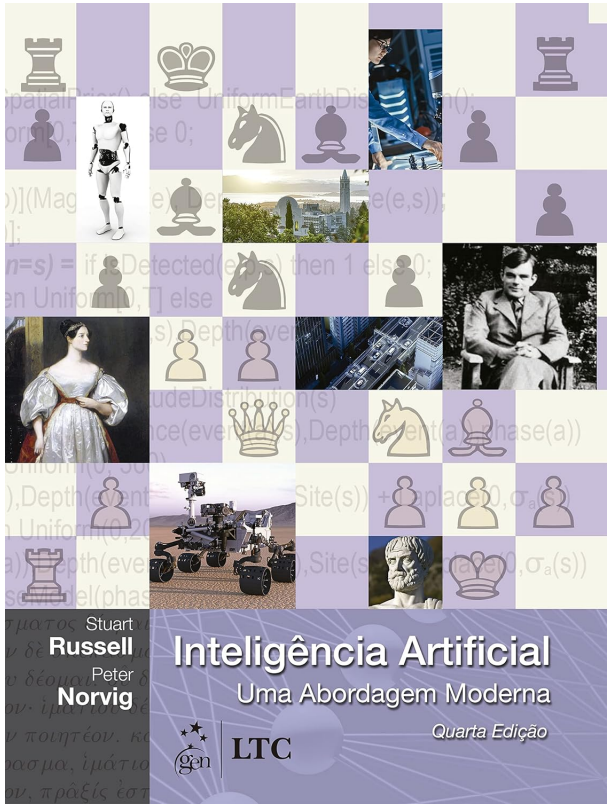
Como avaliar a **qualidade** de uma heurística?

Avaliação de Heurísticas

1. **Precisão**: quão próxima está a estimativa ao custo real?
2. **Custo computacional**: heurística mais elaborada pode ser mais lenta.
3. **Generalização**: heurística construída para um domínio específico pode falhar em outro.

Resumo e Próximos Passos

- Busca informada reduz a complexidade exponencial através de heurísticas.
- **Heurística**: estimativa do custo restante.
  - **Admissível**: nunca superestima.
  - **Consistente**: satisfaz desigualdade triangular; implica admissibilidade.
- **Greedy-Best-First**: expande nó com menor $h(n)$.
- **A***: usa $f(n) = g(n) + h(n)$.
- O projeto da heurística é tão crucial quanto o algoritmo em si.



Atividade recomendada: Leitura do capítulo 3.

Perguntas e Discussão

- Qual é a diferença entre admissibilidade e consistência de uma heurística? Por que a consistência implica admissibilidade?
- Em quais cenários Greedy-Best-First pode superar A* em termos de tempo, apesar de não garantir otimalidade?
- Como o custo computacional de calcular uma heurística mais precisa afeta a eficiência geral do algoritmo?
- Como você justificaria a escolha de uma heurística em um problema real, considerando trade-offs entre precisão e velocidade?
- Em que situações a heurística pode levar o algoritmo a soluções sub-ótimas mesmo sendo admissível?