Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial
Aula 5 - Busca informada, heurística

Prof. José Reinaldo Silva

reinaldo@usp.br

d-l@b

Na aula passada vimos os principais algoritmos de busca não-informada usados em sistemas inteligentes: a busca em profundidade (depth-first), a busca em profundidade limitada (Limited depth-firs), a busca em largura (breadth-first) e busca bidirecional. Vimos, especialmente que os métodos de busca se destacam como uma abordagem simples, direta e confiável para os problemas de engenharia, desde que estes possam ser modelados de forma completa.

George Pólya
1887-1985

PONTES (2019)

HOLOS
ISSN 1807 - 1600

MÉTODO DE POLYA PARA RESOLUÇÃO DE PROBLEMAS MATEMÁTICOS: UMA PROPOSTA METODOLÓGICA PARA O ENSINO E APRENDIZAGEM DE MATEMÁTICA NA EDUCAÇÃO BÁSICA

E.A.S.PONTES*
Instituto Federal de Alagoas
edelpontes@gmail.com*

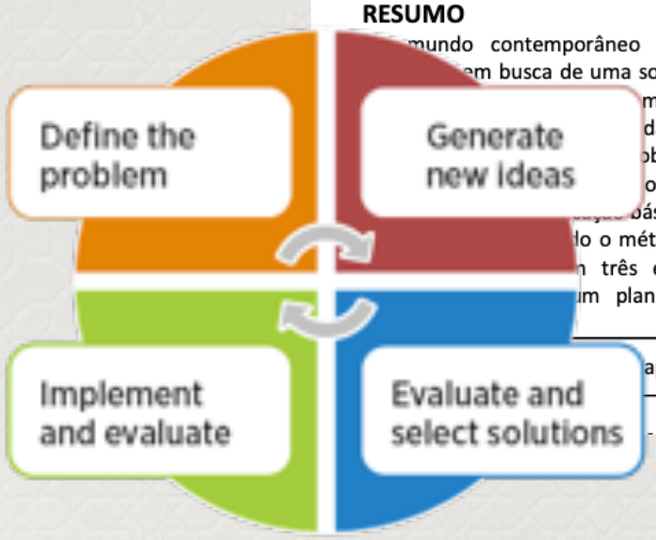
Artigo submetido em 20/12/2017 e aceito em 24/06/2019

DOI: 10.15628/holos.2019.6703

RESUMO

...mundo contemporâneo diversas pesquisas são ...em busca de uma solução eficaz no processo ...matemática, tendo como ...da educação matemática. ...objetivo apresentar uma ...o ensino e aprendizagem ...básica, através da resolução ...lo o método de Polya. O método ...n três etapas: Compreender o ...m plano, Executar o plano e Retrospecto do problema. M... apresentados três problem... resolução seguirá o método ... Resolução de Problemas atrav... como prática educacional no... aprendizagem de matemática... facilitador e ao aluno apre... habilidades no intuito de fo... crítico e o raciocínio lógico.

...aprendizagem de matemática, método de Polya, Resolução d...



IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 13, NO. 6, NOVEMBER/DECEMBER 2001 913

Structured Development of Problem Solving Methods

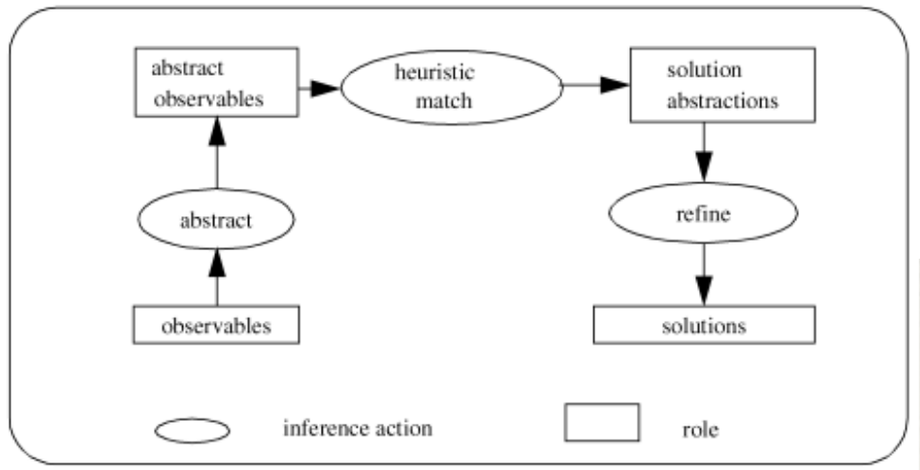
Dieter Fensel and Enrico Motta

Abstract—Problem solving methods (PSMs) describe the reasoning components of knowledge-based systems as patterns of behavior that can be reused across applications. While the availability of extensive problem solving method libraries and the emerging consensus on problem solving method specification languages indicate the maturity of the field, a number of important research issues are still open. In particular, very little progress has been achieved on foundational and methodological issues. Hence, despite the number of libraries which have been developed, it is still not clear what organization principles should be adopted to construct truly comprehensive libraries, covering large numbers of applications and encompassing both task-specific and task-independent problem solving methods. In this paper, we address these "fundamental" issues and present a comprehensive and detailed framework for characterizing problem solving methods and their development process. In particular, we suggest that PSM development consists of introducing assumptions and commitments along a three-dimensional space defined in terms of *problem-solving strategy, task commitments*, and *domain (knowledge) assumptions*. Individual moves through this space can be formally described by means of *adapters*. In the paper, we illustrate our approach and argue that our architecture provides answers to three fundamental problems related to research in problem solving methods: 1) what is the epistemological structure and what are the modeling primitives of PSMs? 2) how can we model the PSM development process? and 3) how can we develop and organize truly comprehensive and manageable libraries of problem solving methods?

Index Terms—Knowledge modeling, problem-solving methods, ontologies, knowledge engineering, software engineering, formal languages.

1 Introduction

Problem solving methods (PSMs) describe the reasoning... number of papers describing the state of the art in problem... com... behavio... instance... ([56], [92... ized by... "revisio... ce—sche... solving... supporti... edge en... framewo... ([55], [87...



"Realizamos hoje o sonho do século XVIII – 'a era da luz', esse sonho dos sábios esperançosos de que a humanidade iria lançar de seus pulsos as velhas algemas; esse sonho de que a superstição estava a caminho de ser vencida pelos ataques sucessivos da ciência experimental e que, conduzidos pela ciência, caminhávamos para uma concórdia e uma felicidade jamais concebidas. Mas já não podemos alimentar tais esperanças. Temos de começar de novo."

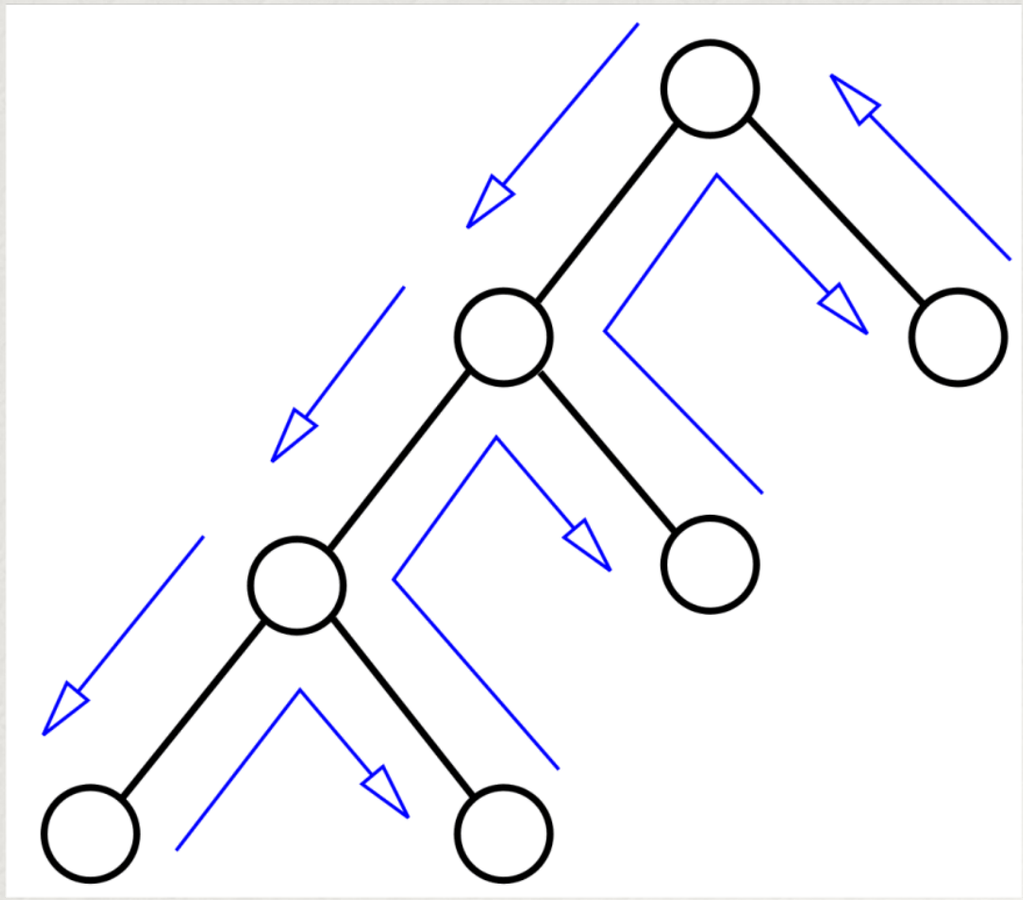
James Harvey Robinson (2022-06-19T22:58:59.000). A Formação da Mentalidade . Montecristo Editora. Kindle Edition.

Busca em profundidade (Depth-first Search, DFS)

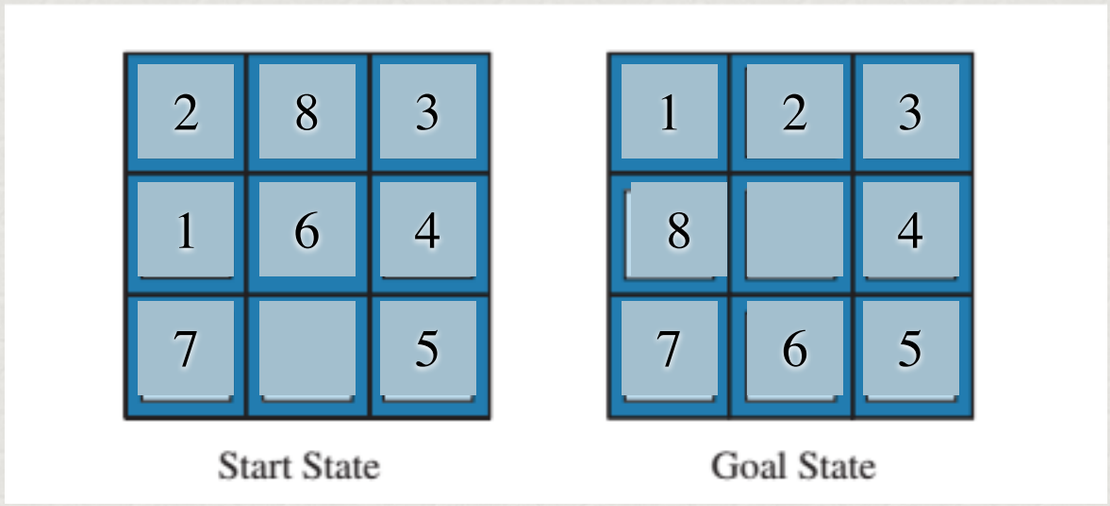
Pilha

```
go(Start, Goal) :-
   empty_stack(Empty_been_list),
   stack(Start, Empty_been_list, Been_list),
   path(Start, Goal, Been_list).

% path implements a depth first search in PROLOG

% Current state = goal, print out been list
path(Goal, Goal, Been_list) :-
    reverse_print_stack(Been_list).

path(State, Goal, Been_list) :-
    mov(State, Next),
    % not(unsafe(Next)),
    not(member_stack(Next, Been_list)),
    stack(Next, Been_list, New_been_list),
    path(Next, Goal, New_been_list), !.

reverse_print_stack(S) :-
    empty_stack(S).
reverse_print_stack(S) :-
    stack(E, Rest, S),
    reverse_print_stack(Rest),
    write(E), nl.
```

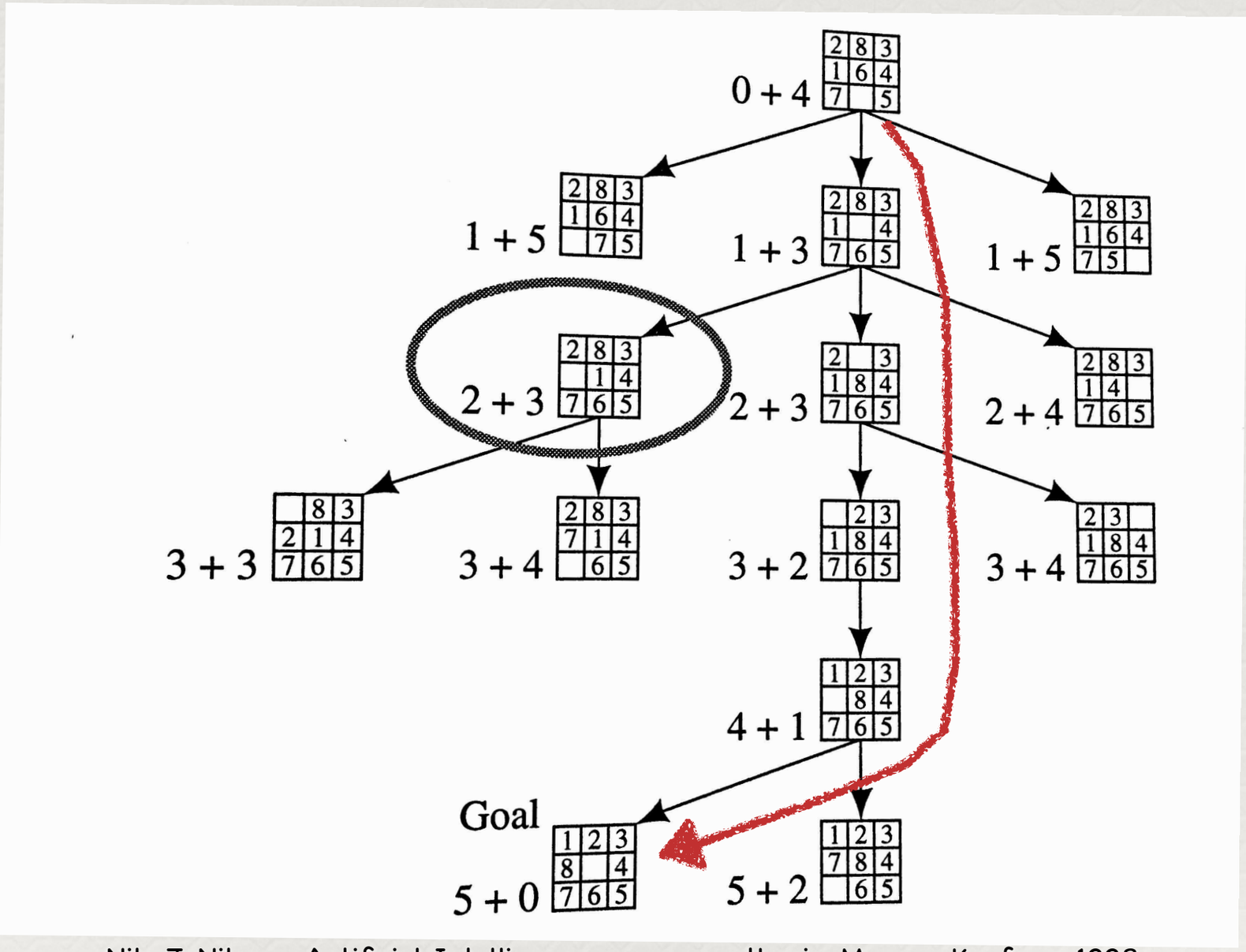
O uso do “stack" é importante para o backtracking.



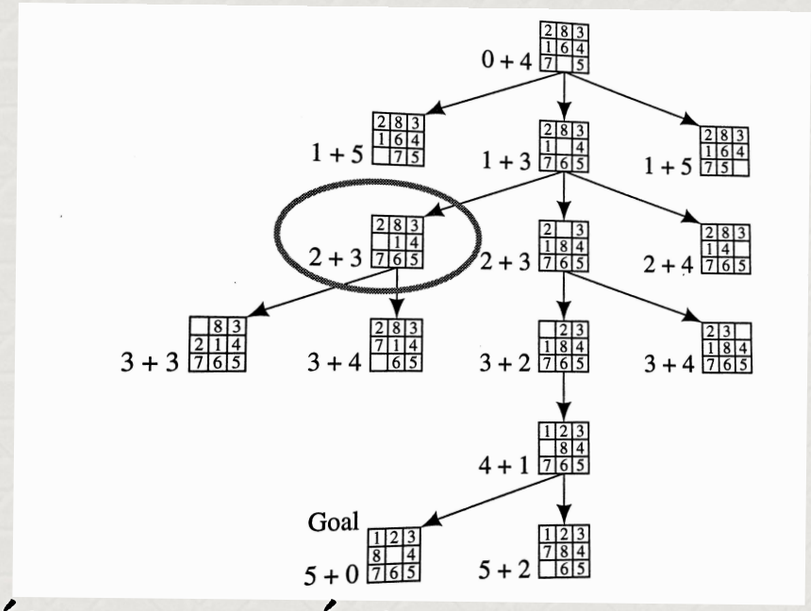
Vamos tomar como exemplo o jogo de tiles com os estados inicial e final mostrados abaixo.



Árvore de busca e a busca informada



Nils J. Nilsson, Artificial Intelligence: a new synthesis, Morgan Kaufann, 1998



A otimização associada a este algoritmo consiste na introdução de uma função que avalia o custo da escolha de cada nó (e da ação leva a este nó). Esta função é denominada <u>função de avaliação f(n)</u>.

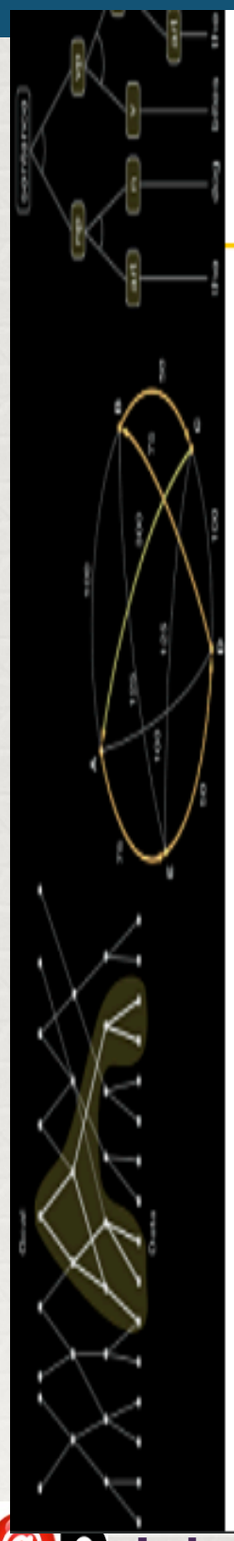
No caso do jogo de tiles esta função é dada por uma heurística h(n) que <u>deve detectar proximidade ao estado final</u>.

Inteligência Artificial

Resolução de Problemas (Parte III)

Prof.ª Joseana Macêdo Fechine Régis de Araújo
joseana@computacao.ufcg.edu.br

Uma heurística **é apenas uma conjectura informada sobre o próximo passo a ser tomado na solução de um problema**. A heurística é baseada na experiência e na intuição. Uma heurística pode levar um algoritmo de busca a uma solução subótima ou, inclusive, levá-lo a não conseguir encontrar uma solução.



Busca Heurística

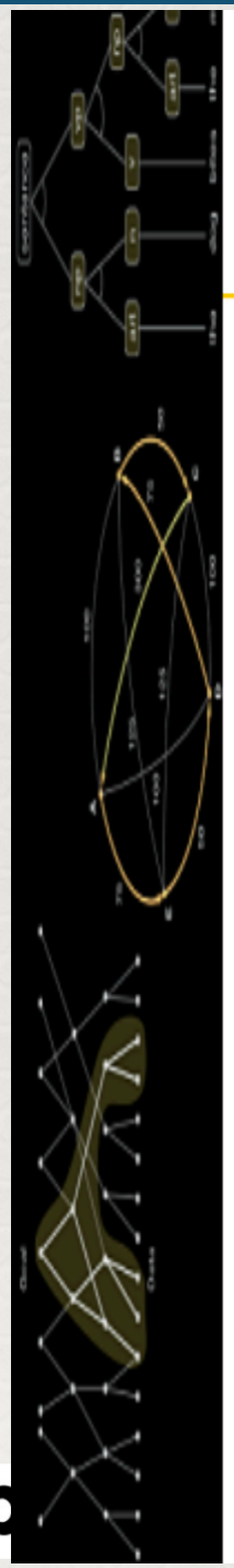
- **Os problemas de IA empregam heurísticas, basicamente, em duas situações**:

1. Um problema pode não ter uma solução exata por causa das ambiguidades inerentes na sua formulação ou pela disponibilidade dos dados.

   Exemplos: Diagnóstico médico, Sistemas de visão.

2. Um problema pode ter uma solução exata, mas o custo computacional para encontrá-la pode ser proibitivo.

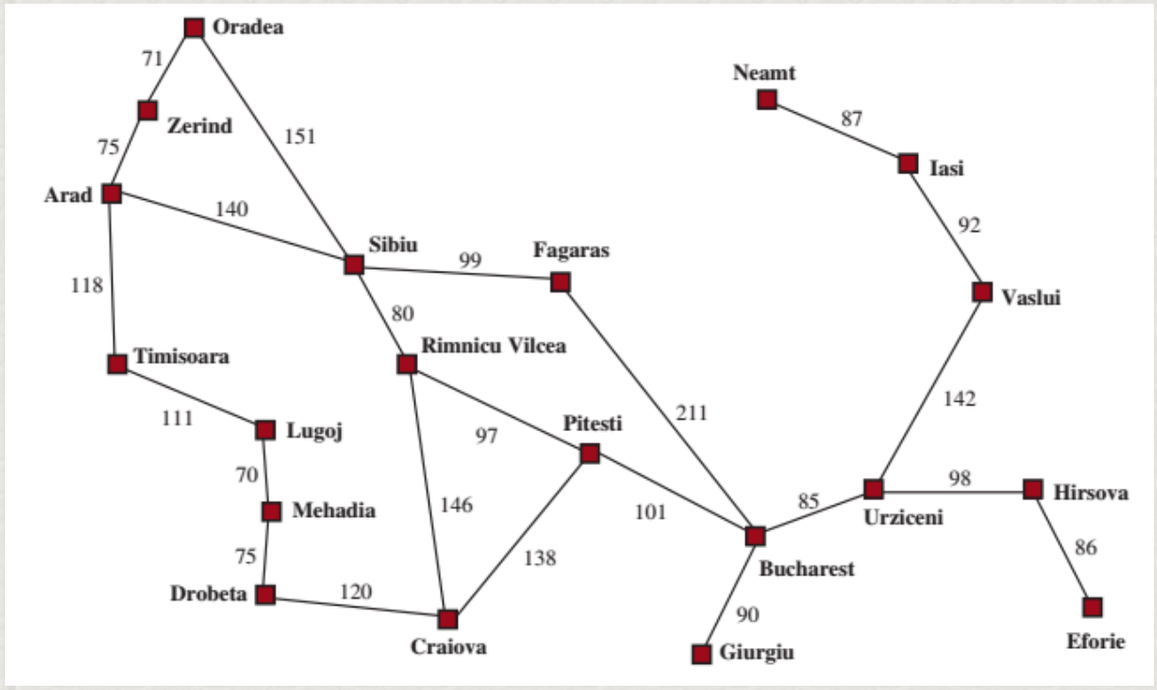
   Exemplo: Jogo de xadrez.



Busca Heurística

- As heurísticas podem falhar.
- Uma heurística é apenas uma conjectura informada sobre o próximo passo a ser tomado na solução de um problema.
- A heurística é baseada na experiência e na intuição.
- Uma heurística pode levar um algoritmo de busca a uma solução subótima ou, inclusive, levá-lo a não conseguir encontrar uma solução.

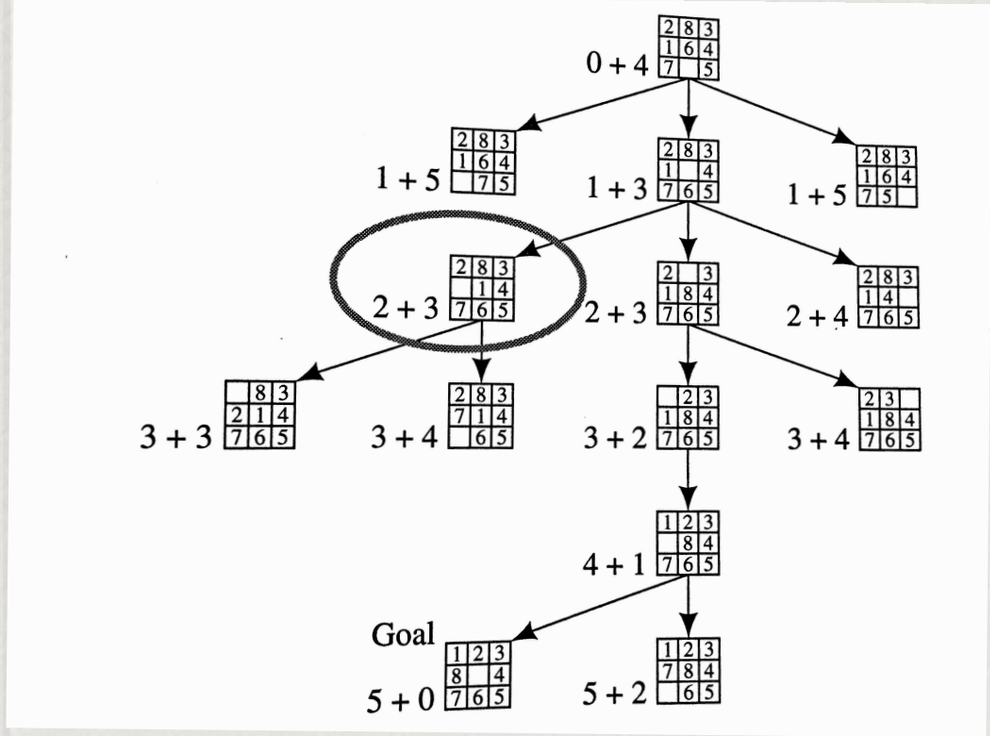
George Polya define heurística como “o estudo dos métodos e das regras de descoberta e invenção” (Polya, 1945) – relacionada com o termo grego original, o verbo eurisco (“Eu descubro”). Quando Arquimedes emergiu de seu famoso banho segurando a coroa de ouro, ele gritou “Eureka!” (“Eu descobri!”).

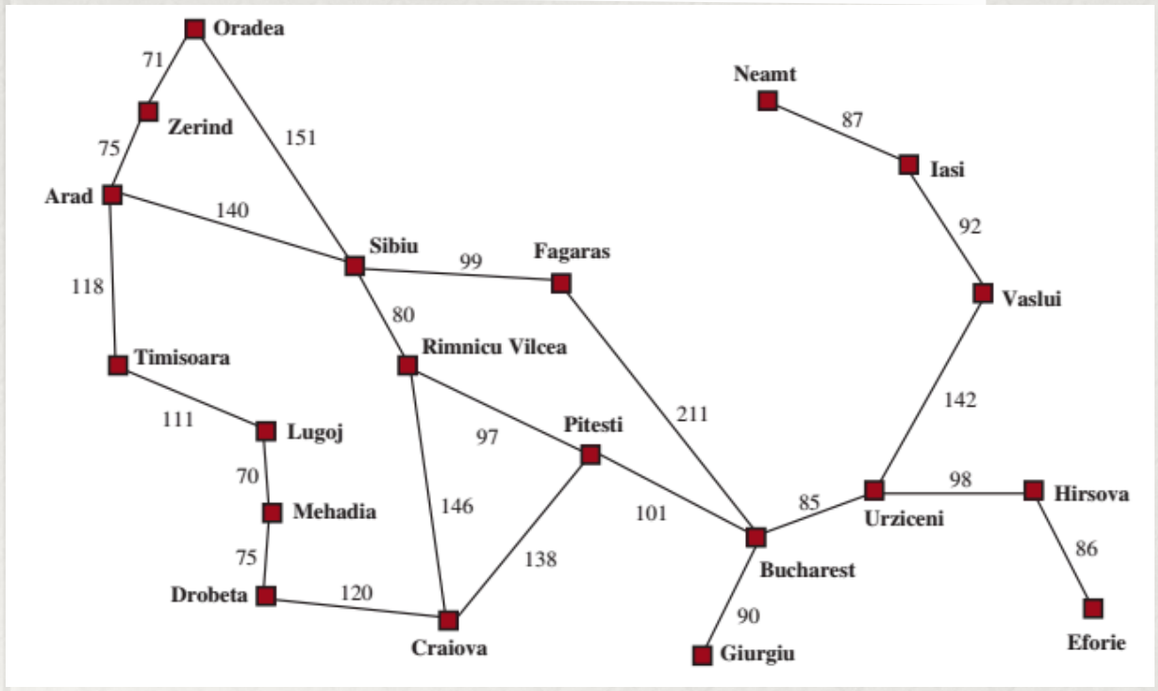


No caso do sistema que busca rotas em uma cidade uma função custo interessante seria a distância em linha reta entre as cidades.

Arad	366	**Mehadia**	241
Bucharest	0	**Neamt**	234
Craiova	160	**Oradea**	380
Drobeta	242	**Pitesti**	100
Eforie	161	**Rimnicu Vilcea**	193
Fagaras	176	**Sibiu**	253
Giurgiu	77	**Timisoara**	329
Hirsova	151	**Urziceni**	80
Iasi	226	**Vaslui**	199
Lugoj	244	**Zerind**	374

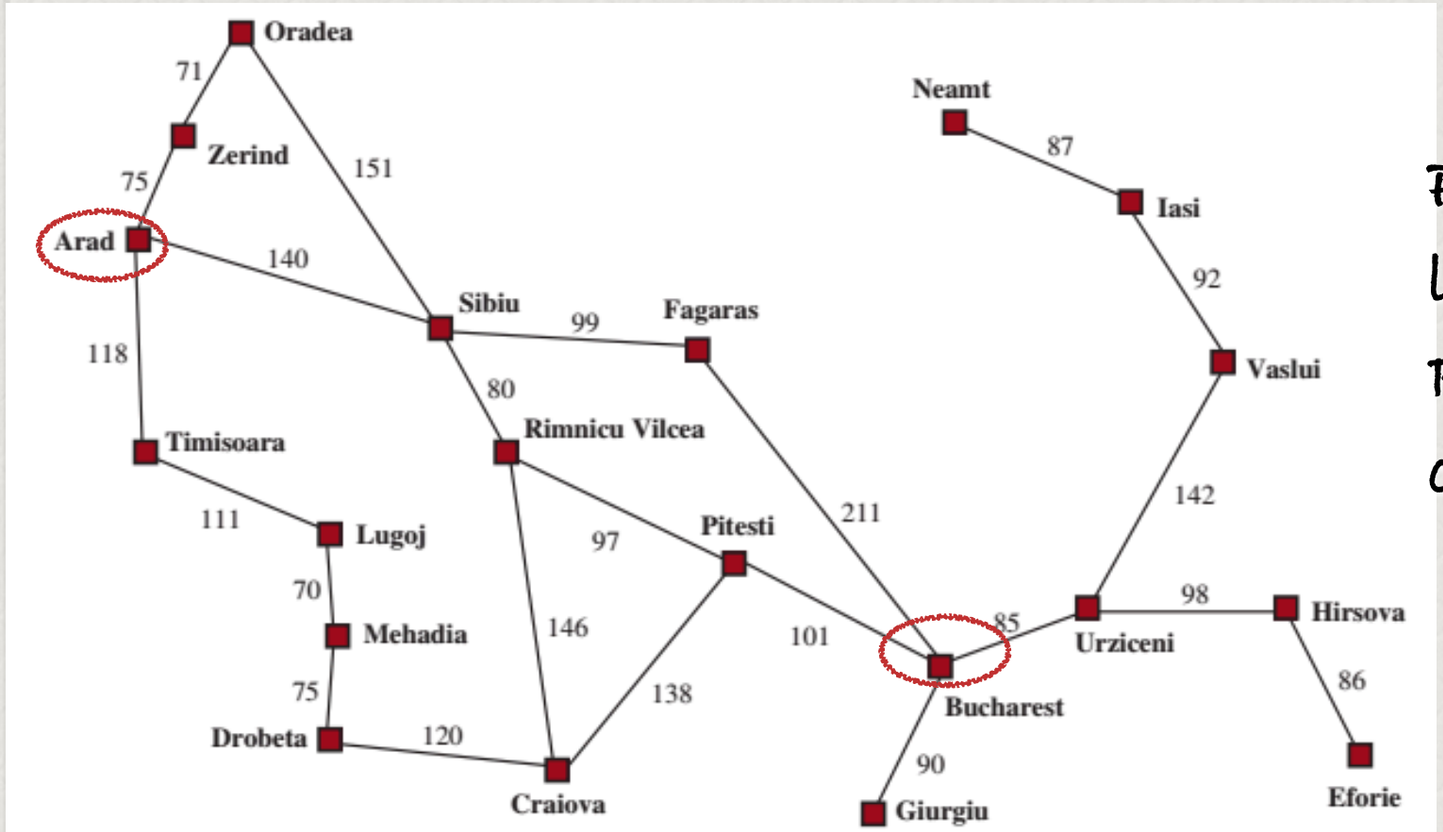
Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.





Greedy best-first

Algoritmos de busca que exploram vizinhança e proximidade ao estado final baseado em uma heurística são chamados greedy best-first.

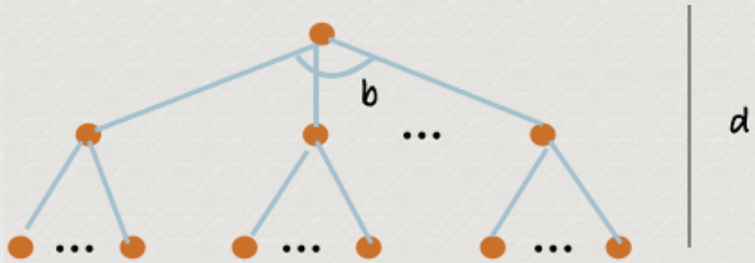


Podemos usar a distância linear de cada cidade do Romênia a Bucharest com critério de escolha da rota.

Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD}—straight-line distances to Bucharest.

No pior caso teremos um algoritmos de complexidade $O(|V|)$, isto é, se tivermos que visitar todos os vértices da árvore. Uma boa heurística pode reduzir isso para $O(b^d)$, onde b é o grau e d a profundidade da árvore.



$$S = (b^d - 1)/(b - 1) \sim O(b^d)$$

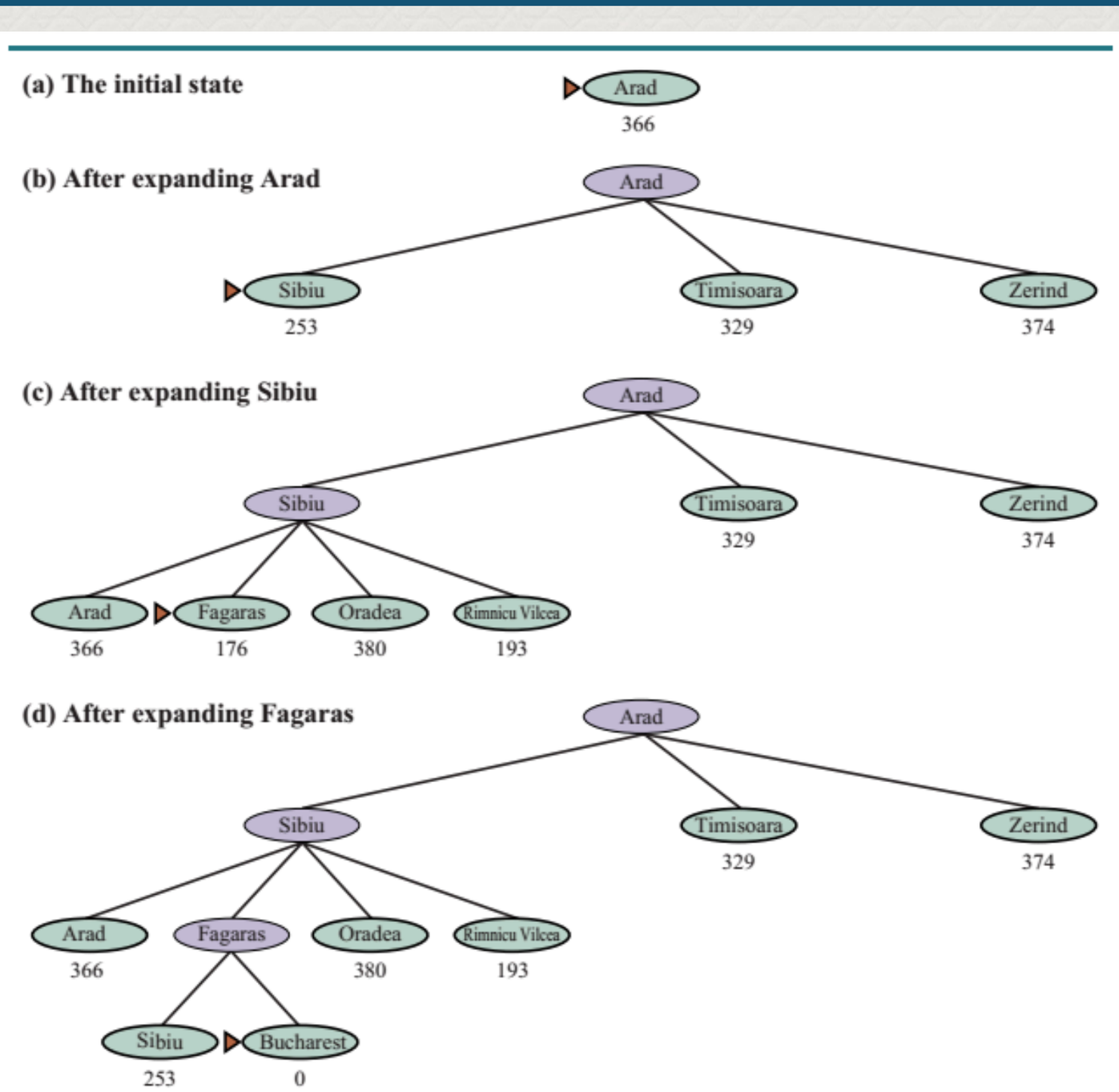


Figure 3.17 Stages in a greedy best-first tree-like search for Bucharest with the straight-line distance heuristic h_{SLD}. Nodes are labeled with their h-values.

O algoritmo A*

Generalizando, podemos ter uma função de avaliação que associa uma função custo g(n) e uma heurística h(n) (eventualmente a proximidade do estado final). Nesse caso

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

No exemplo do jogo de tiles g(n)=p, onde p era o nível de profundidade do elemento visitado na árvore de busca.

Heurísticas

Uma heurística é dita admissível se, para cada nó, esta retorna um valor sempre menor ou igual á "proximidade real" ao estado final.

No exemplo do jogo de tiles h(n) é sempre menor ou igual ao número de movimentos necessários para chegar ao estado final.
No caso do algoritmo de rotas, a distância em linha reta é sempre menor ou igual à distância real seguindo a estrada entre as cidades.

A slightly stronger property is called **consistency**. A heuristic $h(n)$ is consistent if, for every node n and every successor n' of n generated by an action a, we have:

$$h(n) \leq c(n,a,n') + h(n') .$$

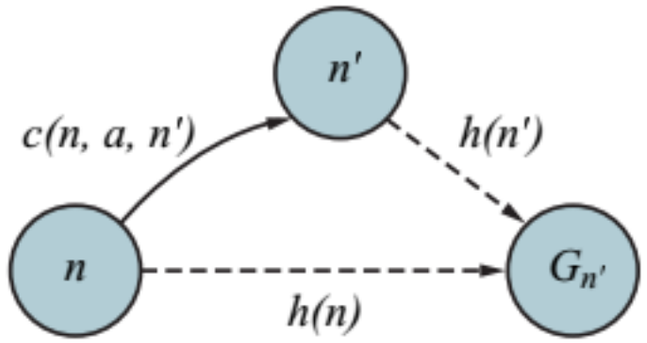
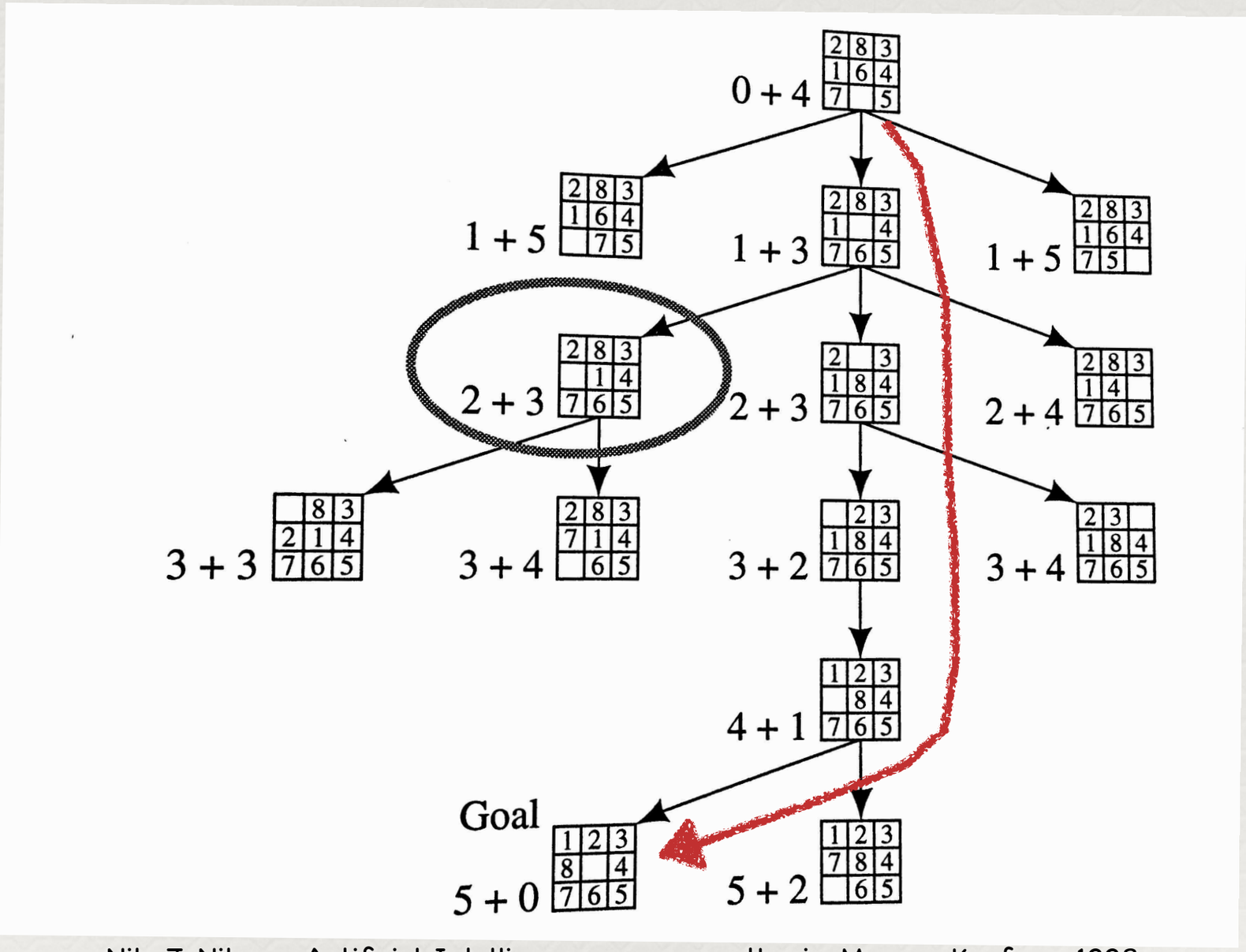


Figure 3.19 Triangle inequality: If the heuristic h is **consistent**, then the single number $h(n)$ will be less than the sum of the cost $c(n,a,a')$ of the action from n to n' plus the heuristic estimate $h(n')$.

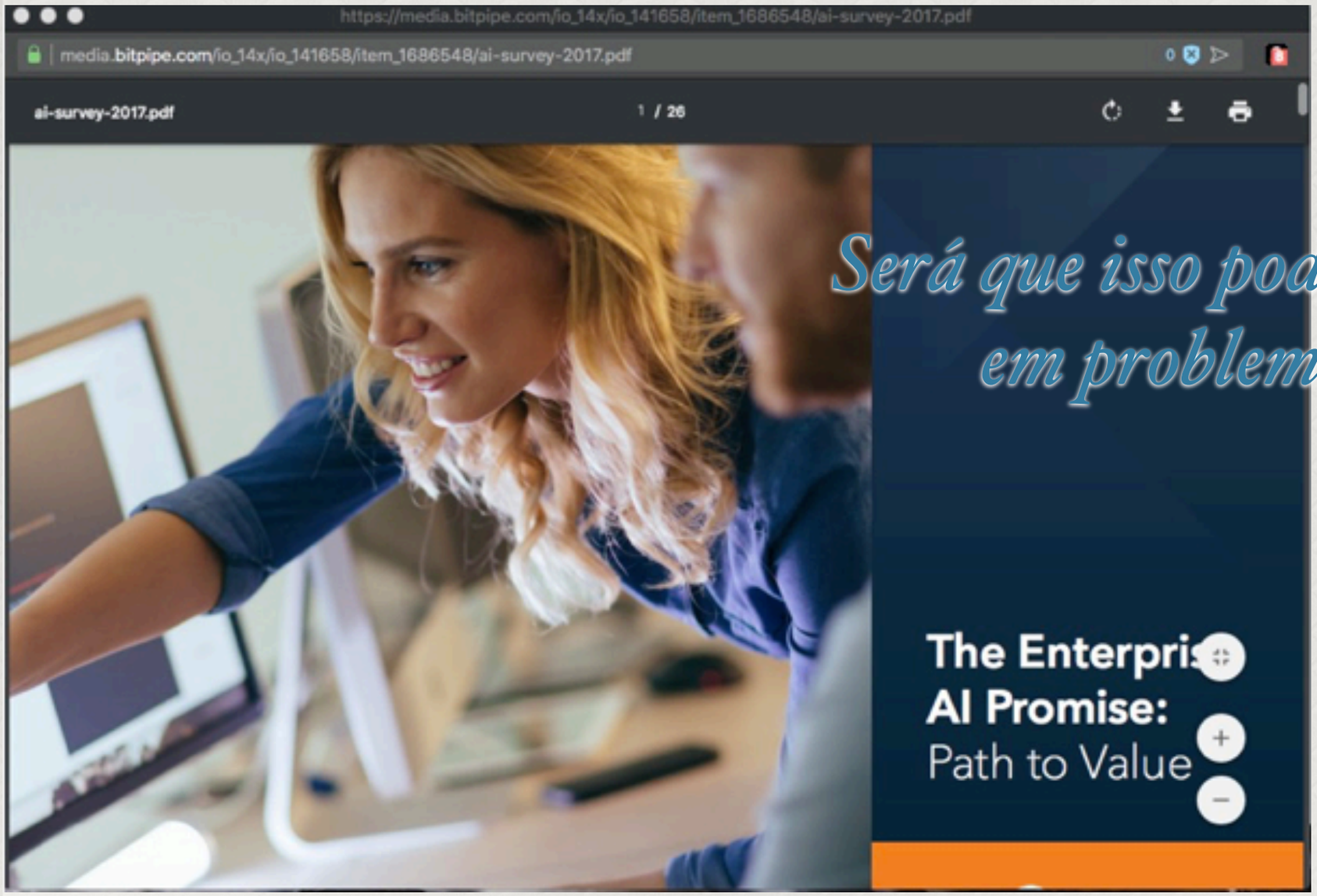
(h(n) é consistente) ⇒ (h(n) é admissível)

Árvore de busca e a busca informada



Nils J. Nilsson, Artificial Intelligence: a new synthesis, Morgan Kaufann, 1998

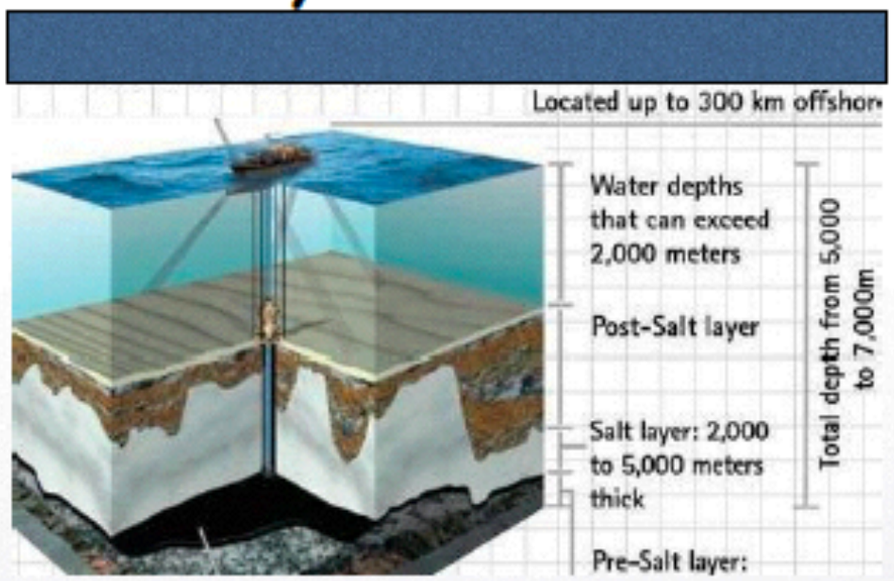
... de olho no mercado!



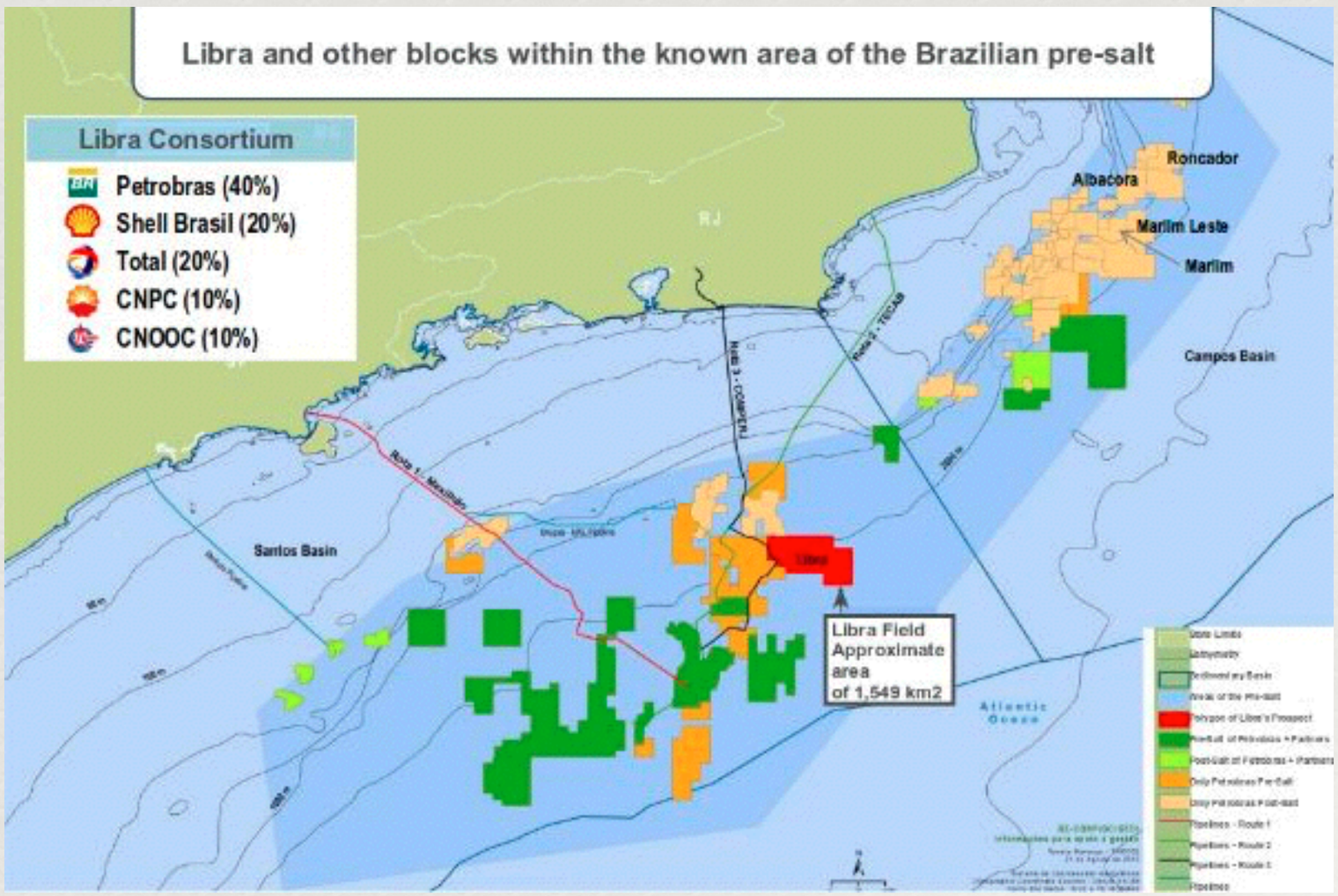
Será que isso pode ser aplicado em problemas reais?

Exemplos: SIPROV (Sistema Inteligente para Programação de Vôo)

Petroleum exploitation in the pre-salt layer



Exemplos: SIPROV (Sistema Inteligente para Programação de Vôo)



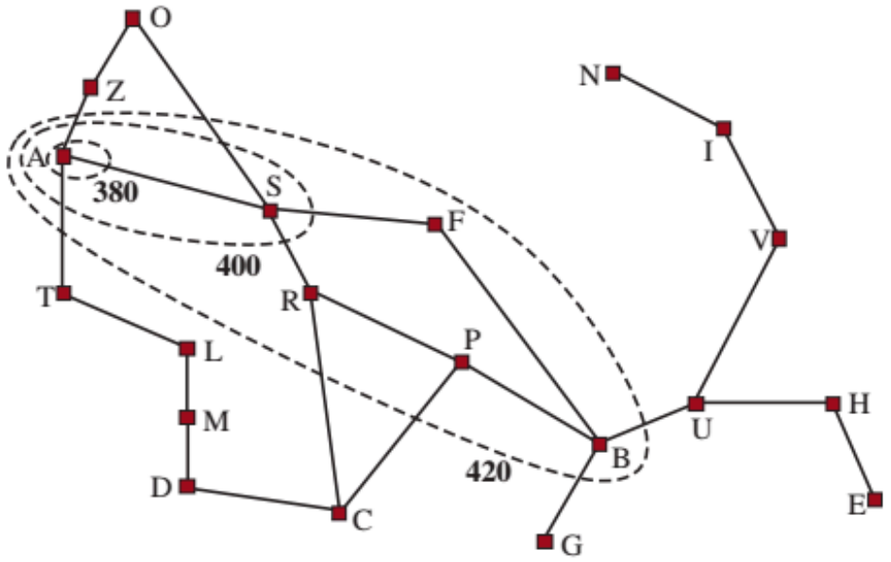
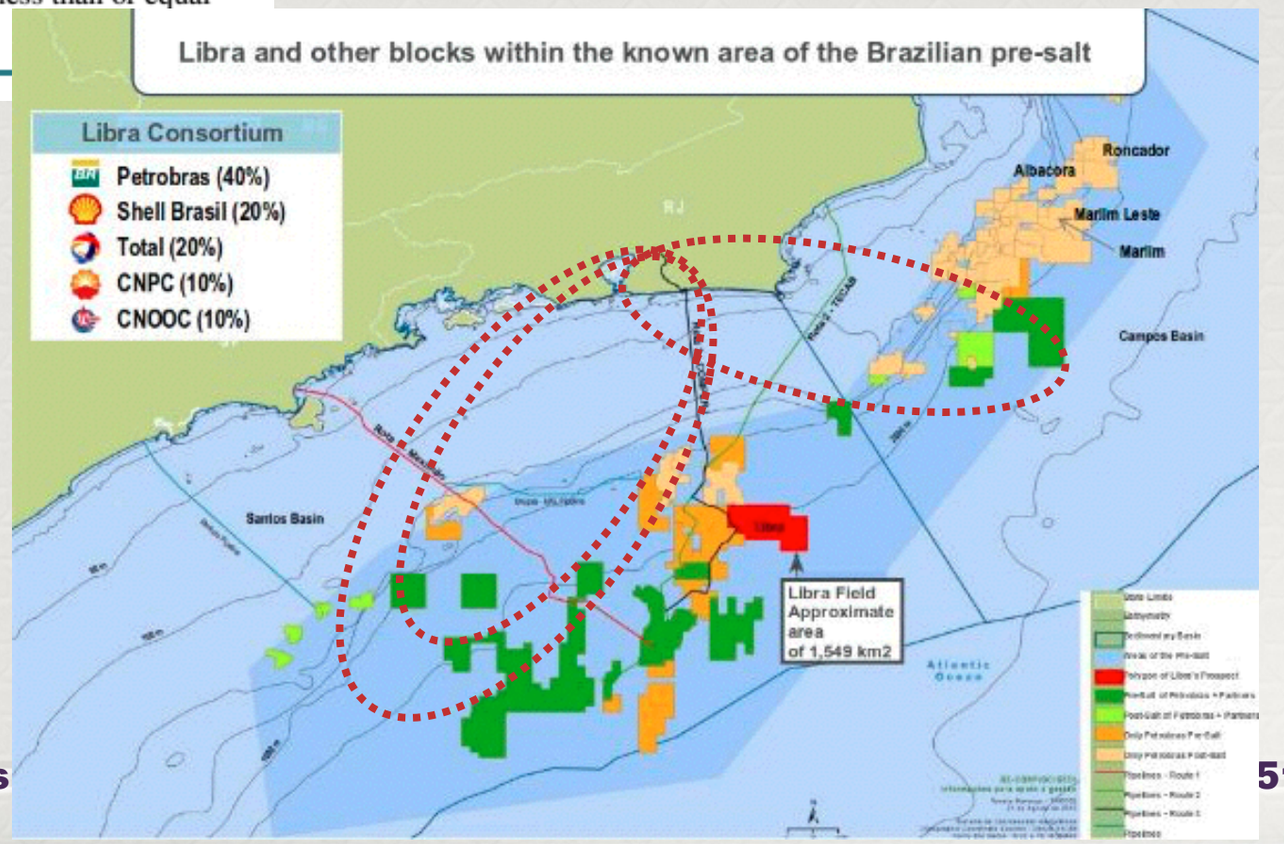


Figure 3.20 Map of Romania showing contours at $f = 380$, $f = 400$, and $f = 420$, with Arad as the start state. Nodes inside a given contour have $f = g + h$ costs less than or equal to the contour value.



Soluções quase-ótimas

Para problemas de grande porte, uma solução estrita para o A* (baseada em ume função de avaliação estrita, e em heurísticas admissíveis e consistentes) pode levar a um coeficiente de expansão muito grande e a um número elevado de nós na árvore de busca. Uma possibilidade é estabelecer um peso que flexibilize a heurística h(n),

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

Dando origem a um algoritmos A* ponderado (com peso)

A* search:	$g(n)+h(n)$	$(W=1)$
Uniform-cost search:	$g(n)$	$(W=0)$
Greedy best-first search:	$h(n)$	$(W=\infty)$
Weighted A* search:	$g(n)+W\times h(n)$	$(1<W<\infty)$

Como criar uma "boa heurística"

O maior obstáculo para uma boa performance na busca e também na busca informada é o tamanho do espaço de estados, mesmo para problemas finitos. O jogo aparentemente simples de tiles (8-puzzle) em 181.400 estados. Se fizermos um jogo com grid maior, o 15-puzzle, teremos mais de um trilhão. Portanto devemos optar por não gerar o espaço de estados previamente, e buscar heurísticas mais fortes, talvez a soma de várias $h_i(n)$.

Usando novamente o 8-puzzle como exemplo, podemos melhorar a heurística considerando, além do número de tiles fora do lugar a distância Manhattan, isto é, quantos movimentos seriam necessários para por o tile no lugar.

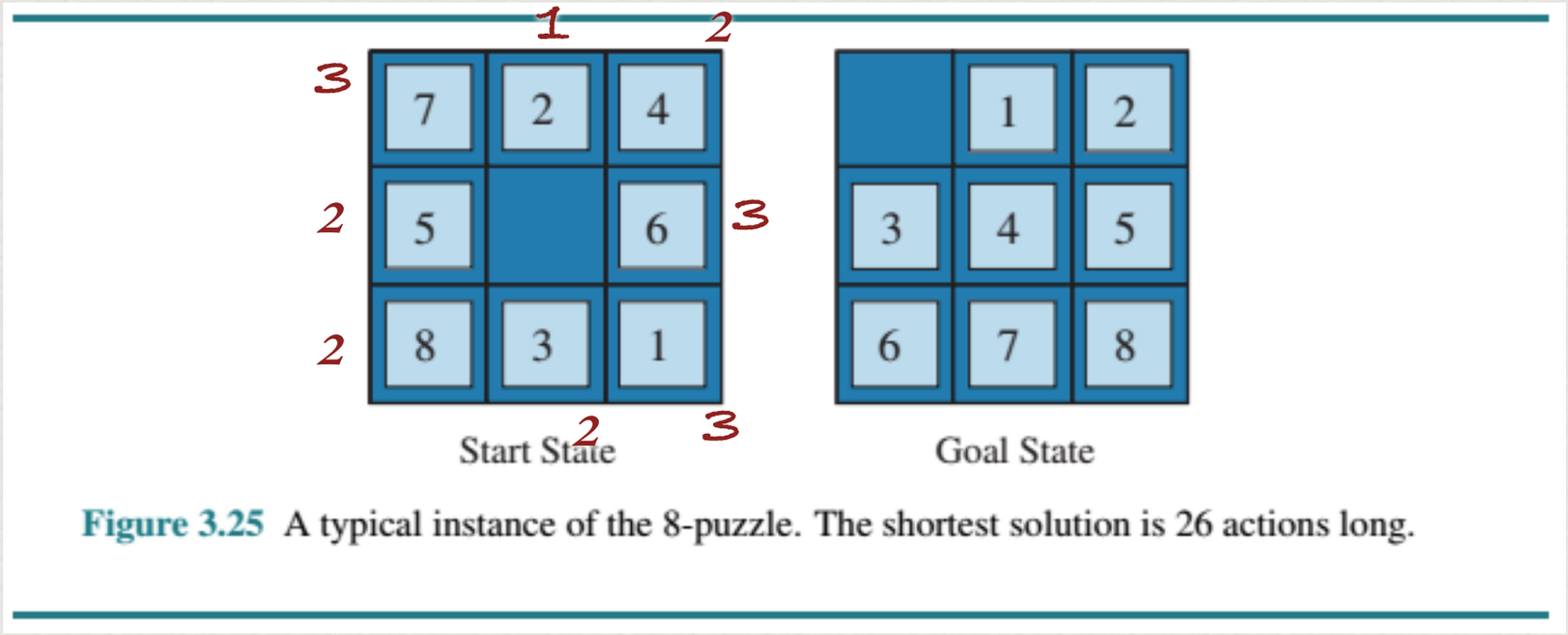
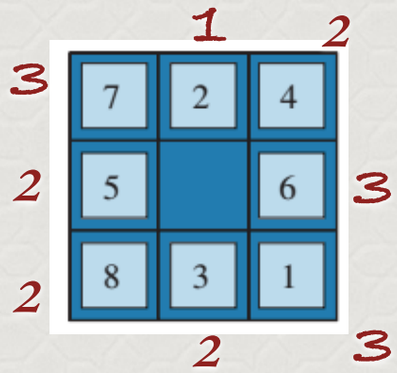
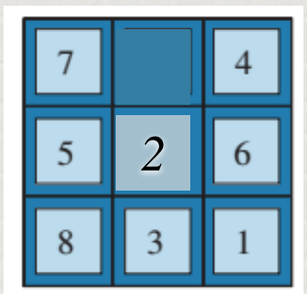


Figure 3.25 A typical instance of the 8-puzzle. The shortest solution is 26 actions long.

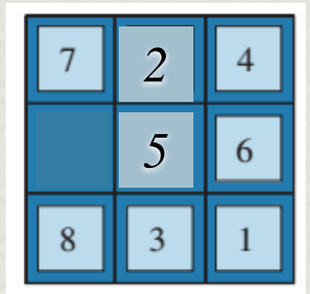


1



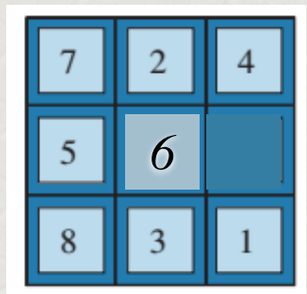
$h_1(1)+h_2(1)=8+19$

2



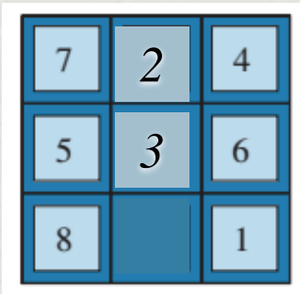
$h_1(1)+h_2(1)=8+17$

3



$h_1(1)+h_2(1)=8+17$

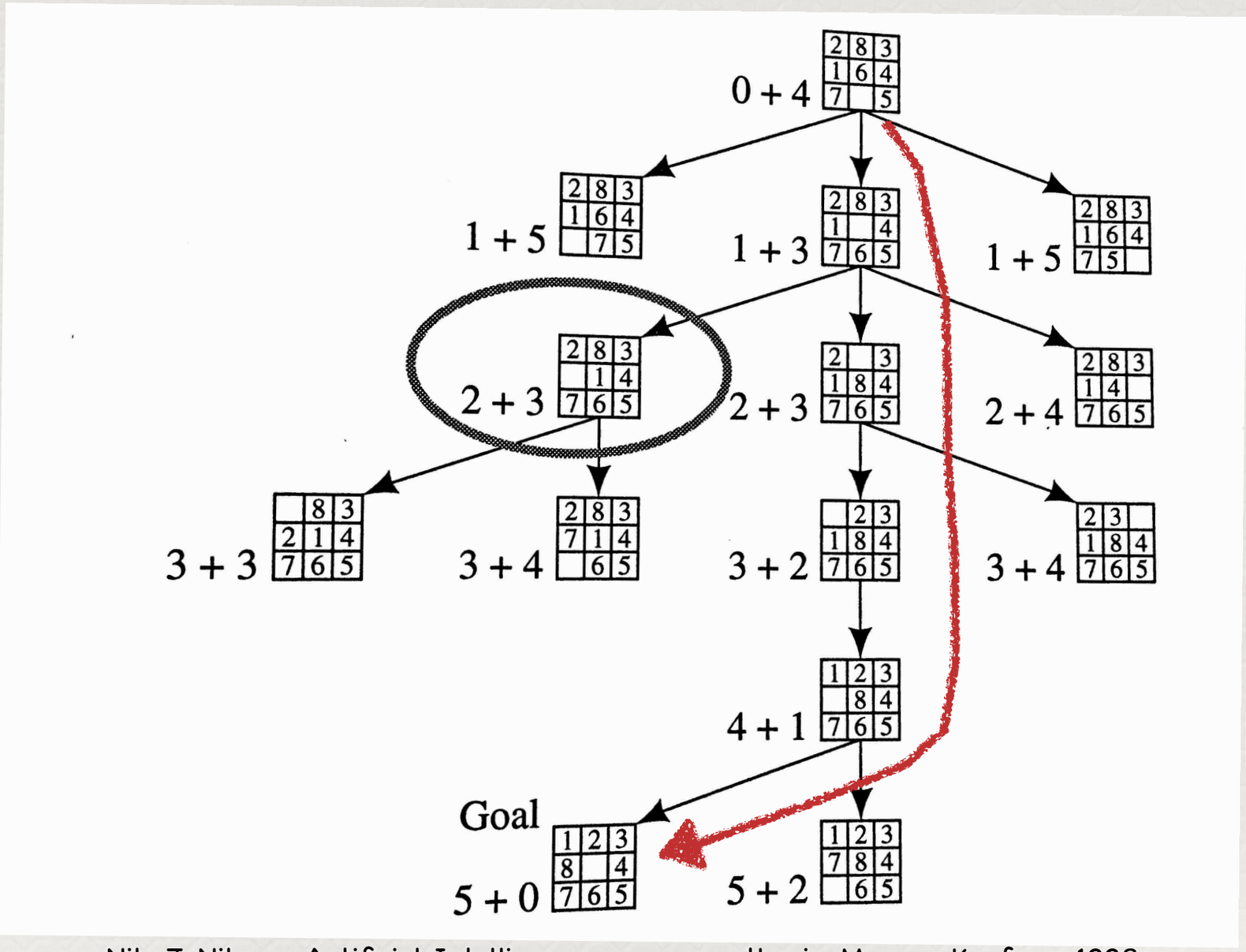
4



$h_1(1)+h_2(1)=8+17$

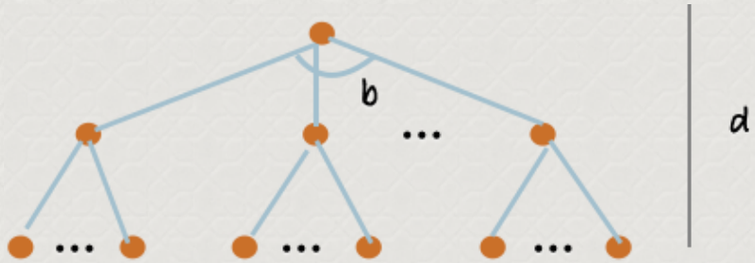
Goal State

Árvore de busca e a busca informada



Nils J. Nilsson, Artificial Intelligence: a new synthesis, Morgan Kaufann, 1998

O maior impacto provocado pela heurística é afetar o expoente d (a profundidade) no processo de busca de um fator k.



$$S = (b^d - 1)/(b - 1) \sim O(b^d)$$

	Search Cost (nodes generated)			Effective Branching Factor		
d	BFS	A*(h_1)	A*(h_2)	BFS	A*(h_1)	A*(h_2)
6	128	24	19	2.01	1.42	1.34
8	368	48	31	1.91	1.40	1.30
10	1033	116	48	1.85	1.43	1.27
12	2672	279	84	1.80	1.45	1.28
14	6783	678	174	1.77	1.47	1.31
16	17270	1683	364	1.74	1.48	1.32
18	41558	4102	751	1.72	1.49	1.34
20	91493	9905	1318	1.69	1.50	1.34
22	175921	22955	2548	1.66	1.50	1.34
24	290082	53039	5733	1.62	1.50	1.36
26	395355	110372	10080	1.58	1.50	1.35
28	463234	202565	22055	1.53	1.49	1.36

Figure 3.26 Comparison of the search costs and effective branching factors for 8-puzzle problems using breadth-first search, A* with h_1 (misplaced tiles), and A* with h_2 (Manhattan distance). Data are averaged over 100 puzzles for each solution length d from 6 to 28.

Resumindo...

Como discutimos na aula passada, a aplicação de algoritmos de busca podem contribuir para agentes que "resolvem problemas" de forma direta, sem uso de "inteligência" (métodos racionais cognitivos ou confeccionistas), na descrição genérica apresentada na nossa segunda aula...

...mas, usando busca informada, e usando heurísticas cada vez mais sofisticadas, ou ainda, relaxando admissibilidade e aceitando soluções próximas do ótimo, que convergem mais rapidamente, começamos a identificar algo que também se pode chamar de "inteligência". Este processo de resolução de problemas é mais propriamente associado à aplicação em sistema de agentes inteligentes.

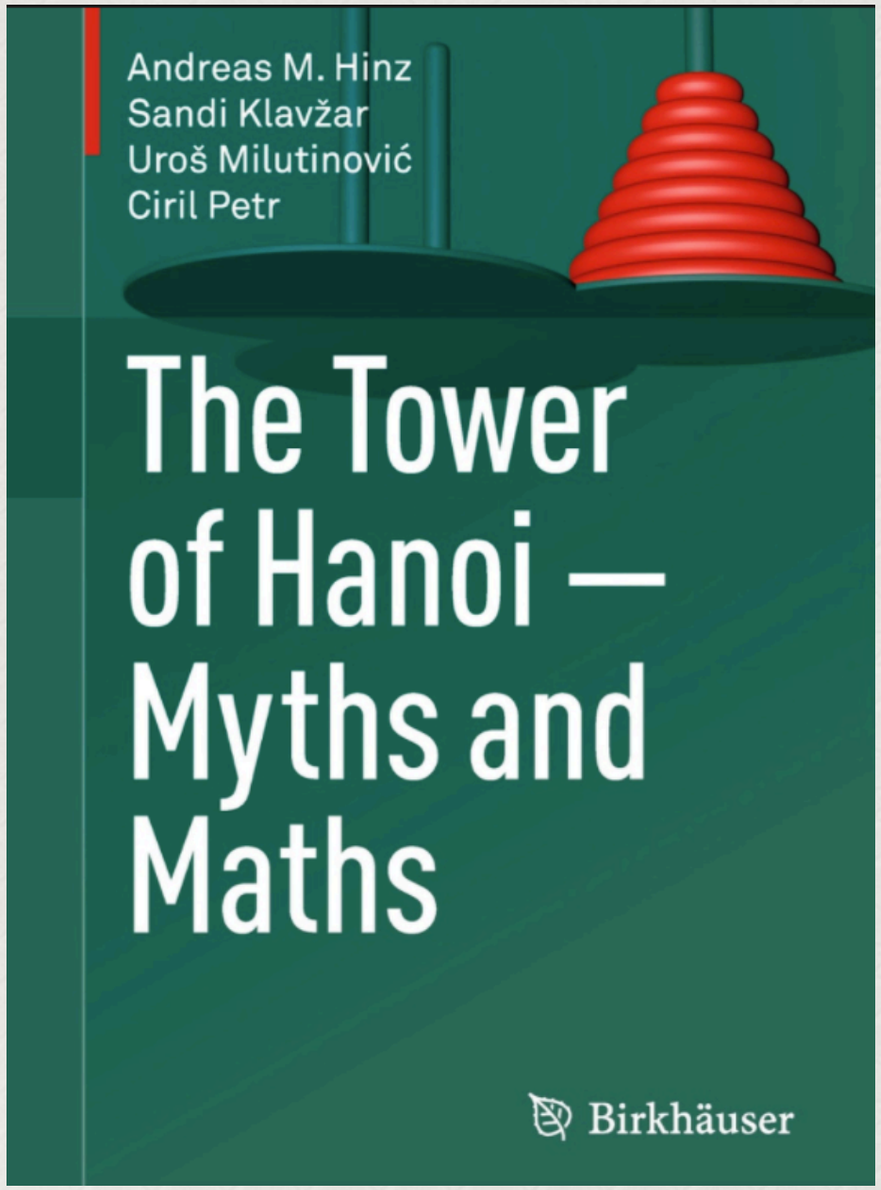
Por outro lado, o que era simplesmente um problema de "programação" de algoritmos de busca passa a ser também um problema de "desenvolvimento" de heurísticas, garantindo admissibilidade e consistência ou relaxando convenientemente a otimização.

A Torre do Brahma

Diz a lenda que o Brahma deu aos seus discípulos uma tarefa-desafio para testar sua tenacidade e perseverança: em 3 hastes distintas foram colocados 64 anéis de diâmetro diferente em ordem crescente. Os monges deveriam transferir todos para uma outra haste sem colocar nunca um anel sobre outro de diâmetro menor, e movendo um anel de cada vez.

Édouard Lucas, 1842-1891

Em 1882, o matemático francês Edouard Lucas publicou um artigo entitulado "Récréations Matematiques", re-editado em 1894, após a morte dele. Um dos problemas tratado é o que foi chamado de "Torre de Hanói", para dar uma explicação formal à solução já indicada por Saravasti, uma das concubinas do Brahma.



2.5 Exercises . . . 128

3 Lucas's Second Problem 131
3.1 Irregular to Regular . . . 131
3.2 Irregular to Perfect . . . 136
3.3 Exercises . . . 140

4 Sierpiński Graphs 141
4.1 Sierpiński Graphs With Base 3 . . . 141
4.2 General Sierpiński Graphs . . . 149
4.2.1 Distance Properties . . . 150
4.2.2 Other Properties . . . 155
4.2.3 Sierpiński Graphs as Interconnection Networks . . . 158
4.3 Connections to Topology: Sierpiński Curve and Lipscomb Space . . . 160
4.4 Exercises . . . 163

5 The Tower of Hanoi with More Pegs 165
5.1 The Reve's Puzzle and the Frame-Stewart Conjecture . . . 165
5.2 Frame-Stewart Numbers . . . 170
5.3 Numerical Evidence for The Reve's Puzzle . . . 179
5.4 Even More Pegs . . . 184
5.5 Hanoi Graphs H_p^n . . . 190
5.6 Numerical Results and Largest Disc Moves . . . 200
5.6.1 Path Algorithms . . . 201
5.6.2 Largest Disc Moves . . . 202
5.7 Exercises . . . 209

6 Variations of the Puzzle 211
6.1 What is a Tower of Hanoi Variant? . . . 211
6.2 The Tower of Antwerpen . . . 218
6.3 The Bottleneck Tower of Hanoi . . . 222
6.4 Exercises . . . 226

7 The Tower of London 227
7.1 Shallice's Tower of London . . . 227
7.2 More London Towers . . . 231
7.3 Exercises . . . 238

8 Tower of Hanoi Variants with Oriented Disc Moves 241
8.1 Solvability . . . 241
8.2 An Algorithm for Three Pegs . . . 245
8.3 More Than Three Pegs . . . 251
8.4 Exponential and Sub-Exponential Variants . . . 256
8.5 Exercises . . . 259

Genericamente a Torre de Hanói se caracteriza por ter três hastes onde se colocam n anéis (em uma das hastes) em ordem crescente (de cima para baixo) de diâmetro. O problema é fazer a transferência em um número mínimo de movimentos (ou próximo disso). Portanto o primeiro desafio é saber quantos movimentos são necessários para - no mínimo - mover n anéis para o seu destino.

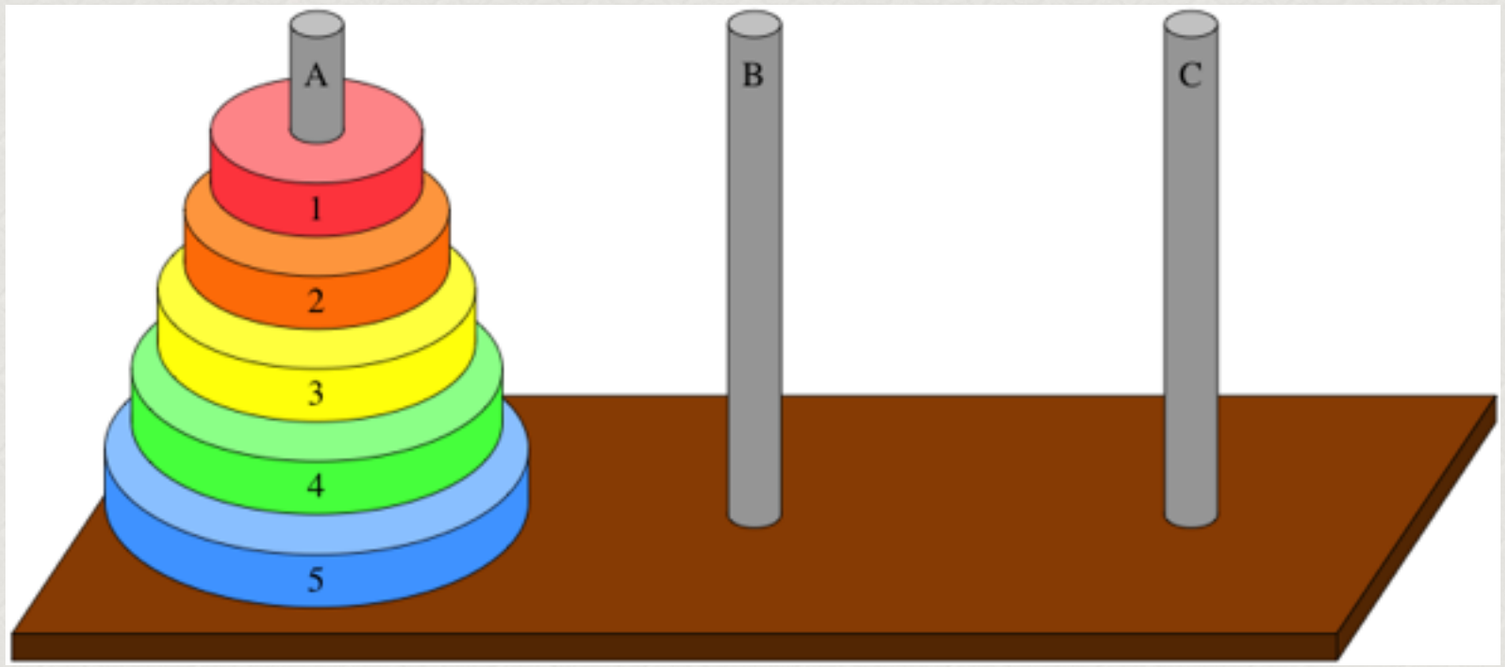
O problema é eminentemente recursivo e por recursão se pode deduzir o número mínimo de movimentos:

$T(1) = 1;$

$T(2) = 3;$

$T(3) = 7;$

$T(n) = 2^n - 1$



A Torre de Hanoi

63 movimentos

O nosso primeiro exercício programa será mais simples este ano: vocês devem implementar usando o swish Prolog, um "jogador" para o 8-puzzle (ou jogo de tiles), que seja melhor do que um jogador que use as heurísticas que acabamos de discutir. Os que fizerem um algoritmo com custo e h1 poderão atingir a nota 5. A partir daí podem ganhar mais pontos, começando com a inclusão da Manhatam. O runtime de cada equipe será computado como se fosse uma competição.

Resolver este problema usando IA, ou, programar um agente inteligente para resolver este problema (qualquer que seja o estado inicial) significa:

i) desenvolver a função de avaliação, função custo e heurística(s);
ii) verificar se a heurística é admissível e consistente e se a função custo é monotônica.
iii) Testar a função de avaliação em um caso simples;
iv) Fazer o programa em SWISH Prolog.

Deadline: Sexta-feira, 7 de outubro.

Os programas devem ter documentação, começando pela identificação da equipe.

Todos devem usar o SWISH Prolog do D-Lab, no link repassado no início do curso (o mesmo onde está o tutorial).

Rahul Casanovas 11762540

- Giulia Duo Cardella 10770392
- Guilherme Rodrigues Monteiro 10706103
- Luiz Fernando Ferreira da Silva 10770603
- Yae Do Choi 8585821

Gabriela Fonseca Fanucchi – 10770371

Guilherme Henrique Martins de Oliveira - 9839008

Yuri Lopes Pamplona - 10853498

João Victor Lins Fregnan NUSP:10333737
Victor Kendy Kamia NUSP: 10791659
Vinícius Araujo da Costa NUSP:7254069

Bernardo Moredo Rocco - 10706145
Victor Benito Rafael Alves Pereira - 10335253
Vinícius Araujo da Costa - 7254069

NUSP	NOME
10333442	Lucas Oliveira Reis
10333654	Pablo Nabil Villar Bou Assi
11741578	Samuel Flávio de Paiva Araujo
11741557	Vinicius Rocha Paiva

Lucas Nascimento Tulha NUSP 10773652

Marcelo Alonso Ronceros Cordova NUSP 14032840

Vidal gonzalo flores rojas NUSP 10773029

Daniel Willian Braz Pires Domingues	N°USP: 10337262
Gabriel Antonio Ken Chang	10770537
Mauricio Hiroki Tomida	10770412
Maykon Souza Cruz	10669490

Perguntas?