Podstawy sztucznej inteligencji

wykład II
Problem solving

Joanna Kołodziejczyk

03 październik 2012

Jakie problemy możemy rozwiązywać?

Cel:

Zbudować inteligentnego agenta planującego, rozwiązującego problem.

- Szachy
- Kostka rubika
- Krzyżówka
- Labirynt
- Wybór trasy

Przykład planowania trasy

Definiowanie problemu

- Stan początkowy *initial state*

Definiowanie problemu

- Stan początkowy *initial state*
- Operator(S) – akcja/funkcja $action(S) \mapsto \{a_1, a_2, \ldots, a_n\}$

Definiowanie problemu

- Stan początkowy *initial state*
- Operator(S) – akcja/funkcja $action(S) \mapsto \{a_1, a_2, \ldots, a_n\}$
- Wynik: $result(S, a) \mapsto S'$

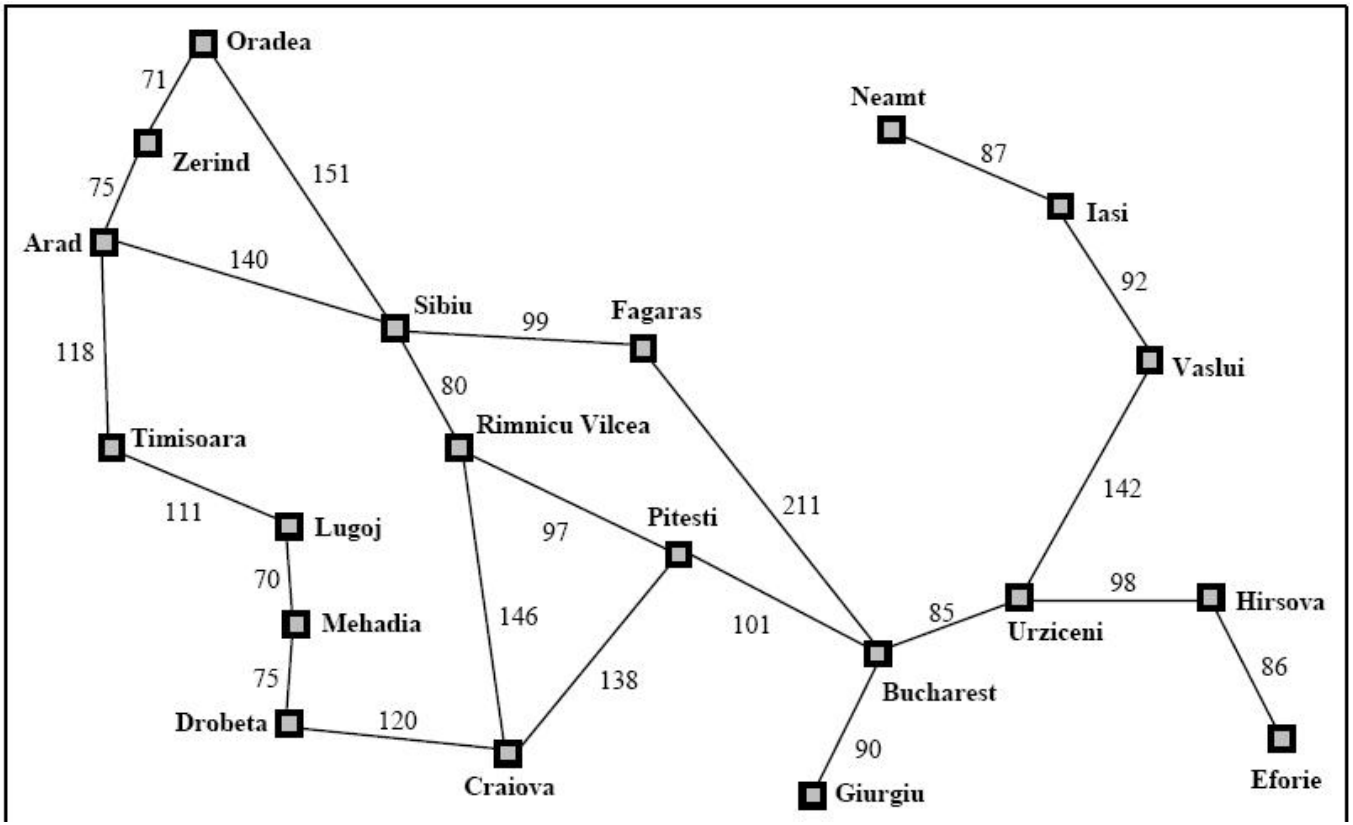
Definiowanie problemu

- Stan początkowy *initial state*
- Operator(S) – akcja/funkcja $action(S) \mapsto \{a_1, a_2, \ldots, a_n\}$
- Wynik: $result(S, a) \mapsto S'$
- Czy wynik?: $goaltest(S) \mapsto True/False$

Definiowanie problemu

- Stan początkowy *initial state*
- Operator(S) – akcja/funkcja $action(S) \mapsto \{a_1, a_2, \ldots, a_n\}$
- Wynik: $result(S, a) \mapsto S'$
- Czy wynik?: $goaltest(S) \mapsto True/False$
- Koszt ścieżki (Funkcja addytywna):
  $pathcost(S \xrightarrow{a_i} S' \xrightarrow{a_j} S'') \mapsto n$
  $stepcost(S, a_i, S') \mapsto n$

Definiowanie problemu na przykładzie planowania trasy



Przeszukiwanie drzewa - funkcja opisująca rodzinę algorytmów

function TREE-SEARCH(problem) returns path

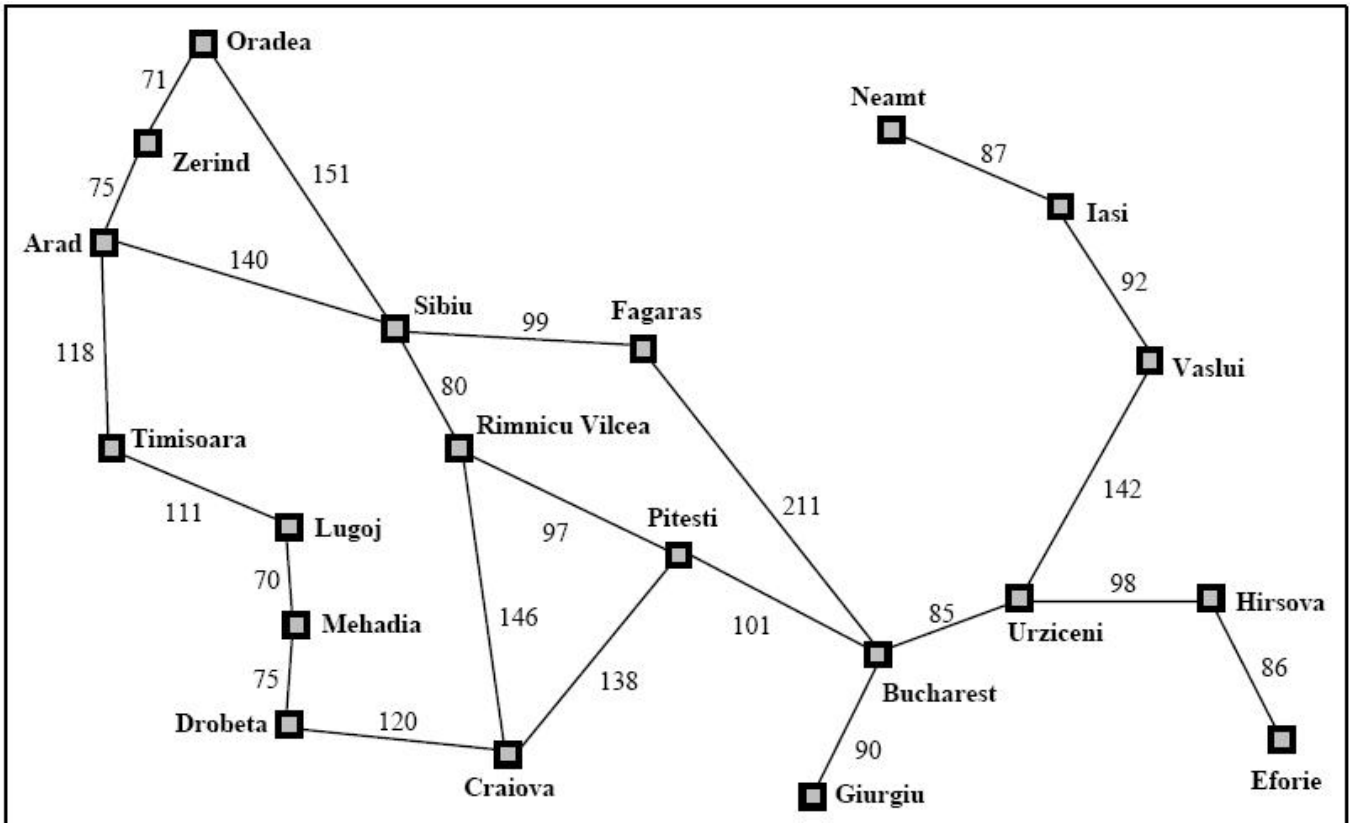
$Frontier = \{path.initial\ state\}$
loop:

if frontier is empty then return FAIL
$path = remove\_choice\{frontier\}$
$S = path.end$
if $goaltest(S)$ then return path
for $a \in Actions(S)$: add $[path + a \mapsto Result(S, a)]$ to frontier

Ostatnia linijka opisuje procedurę rozwinięcia węzła (Stanu).

Przeszukiwanie wszerz dla drzew na przykładzie planowania trasy

Wybiera ze stanów do rozwinięcia zawsze te stany, które są najbliżej.



Przeszukiwanie grafu - funkcja opisująca rodzinę algorytmów

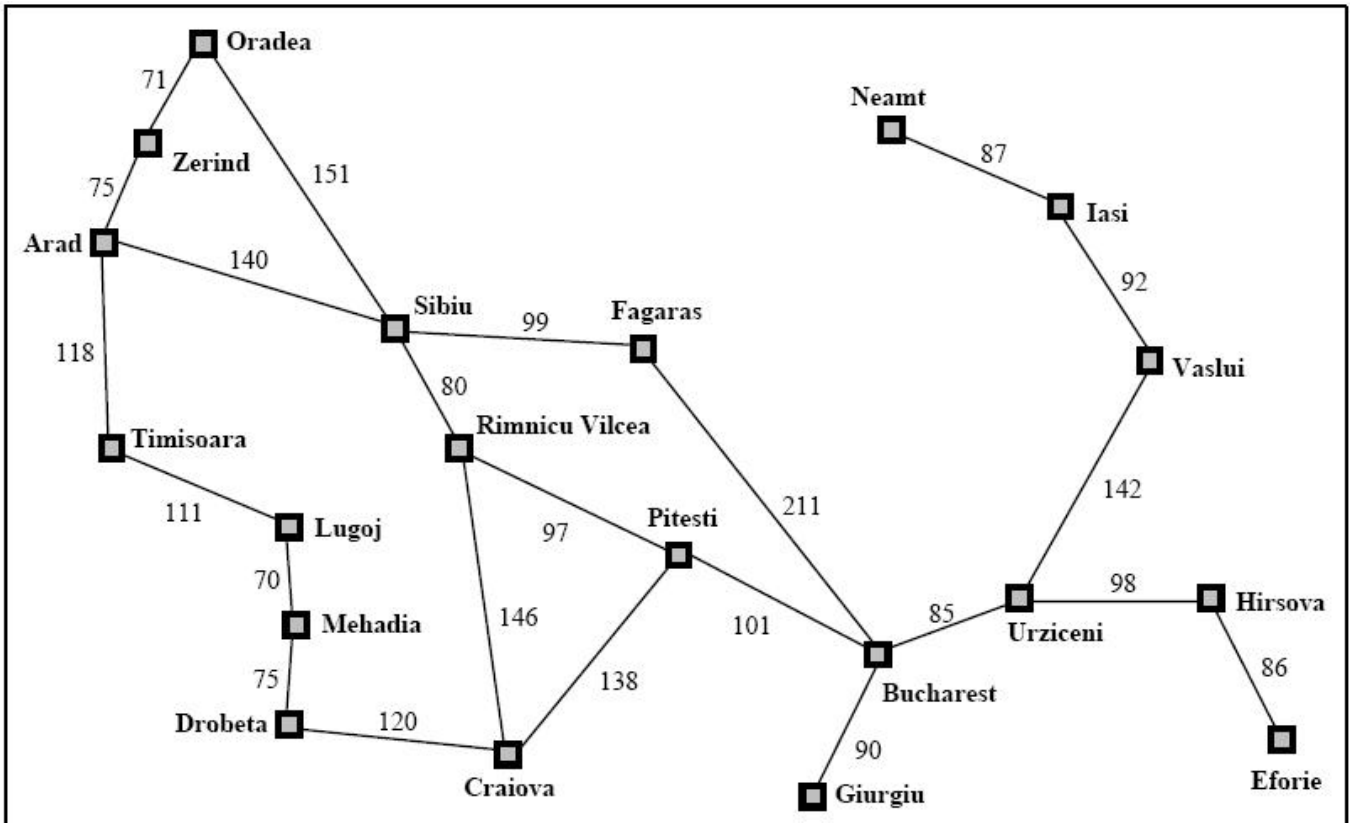
function GRAPH-SEARCH(problem) returns path

$Frontier = \{path.initial\ state\}$, $explored = \{\}$
loop:

if frontier is empty then return FAIL
$path = remove\_choice\{frontier\}$, add S to explored
$S = path.end$
if $goaltest(S)$ then return path
for $a \in Actions(S)$: add $[path + a \mapsto Result(S, a)]$ to frontier
unless $Result(S, a)$ is in frontier or explored

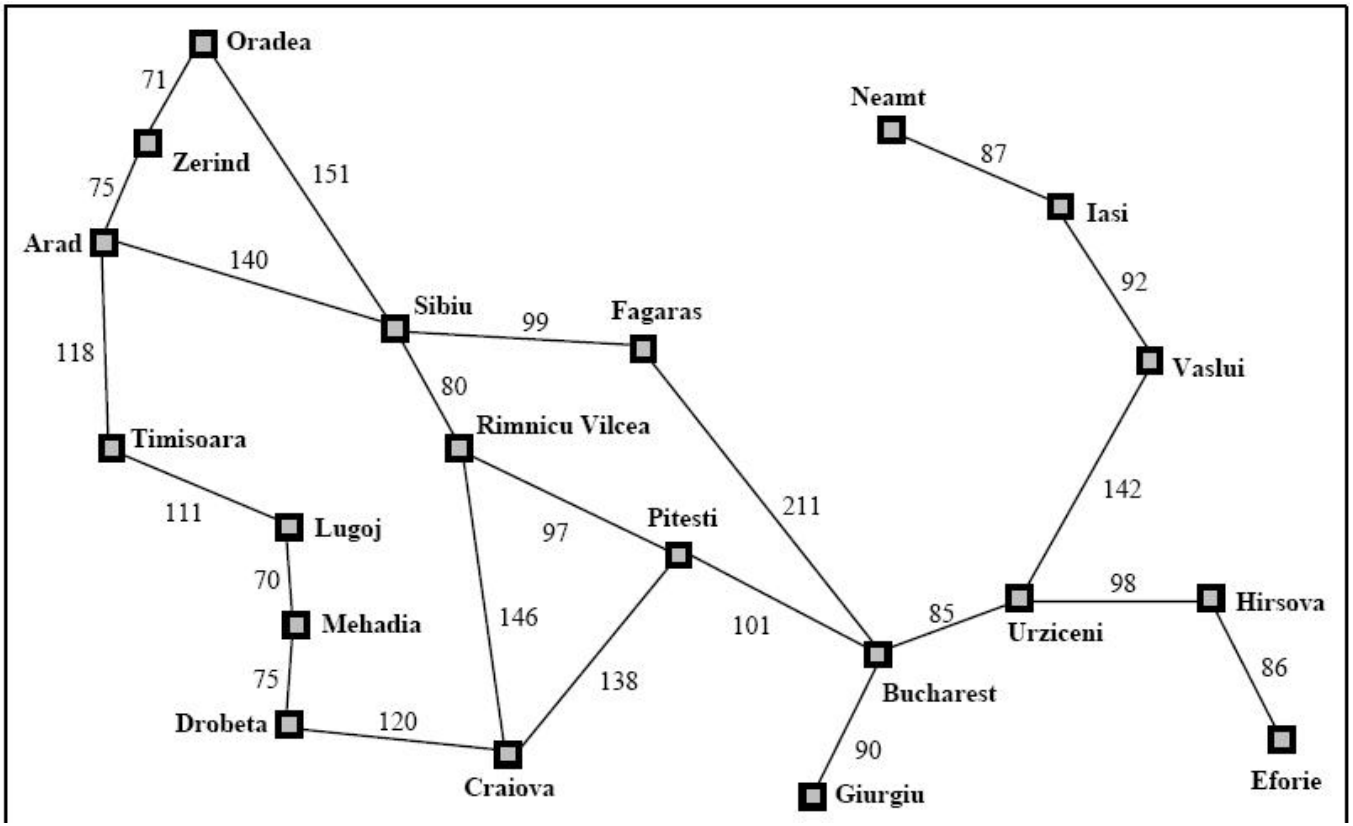
Takie przeszukiwanie unika ponownego sprawdzania węzłów już odwiedzonych.

Przeszukiwanie wszerz dla grafu na przykładzie planowania trasy

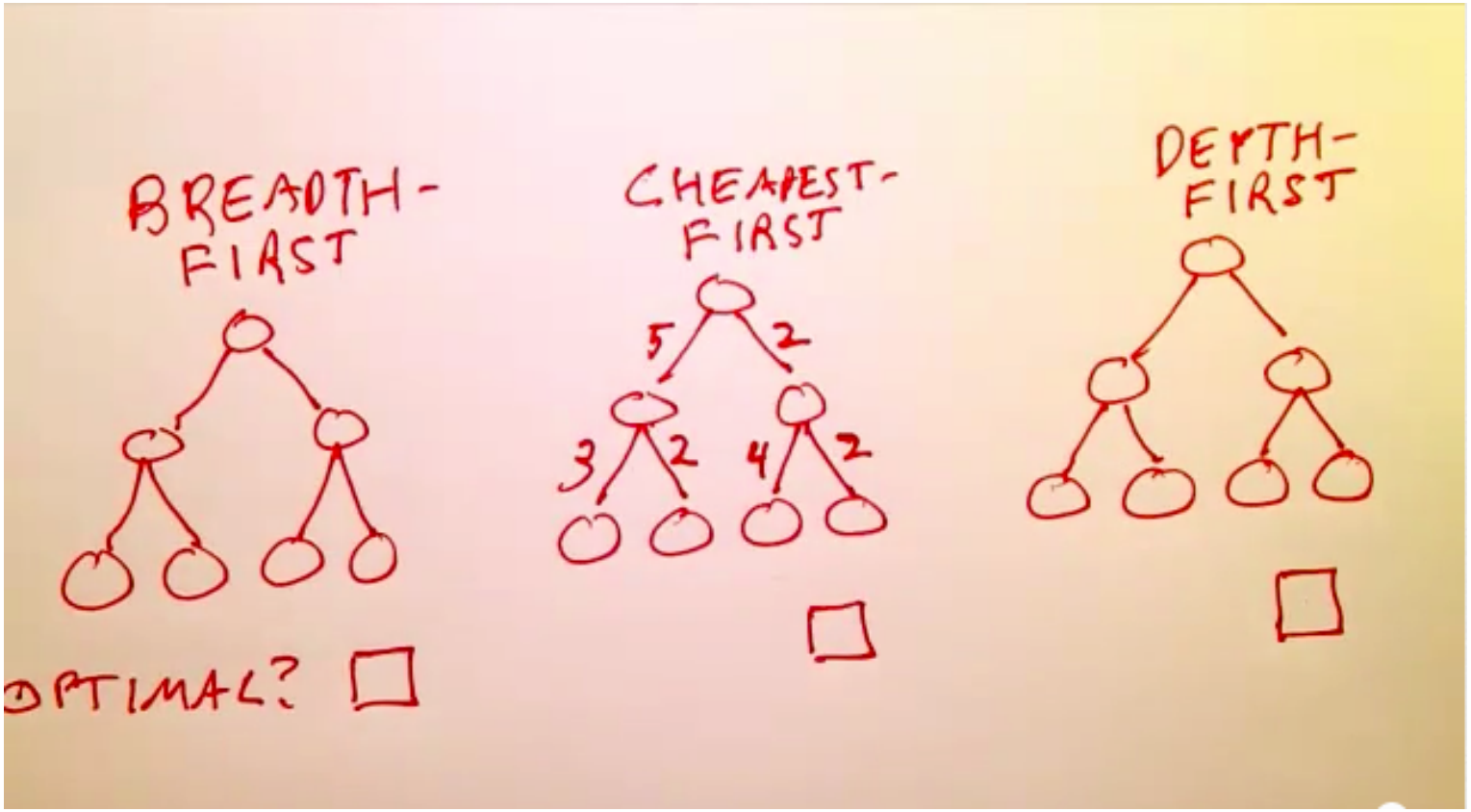


Przeszukiwanie przy stałych kosztach dla grafu na przykładzie planowania trasy

Wybiera ze stanów do rozwinięcia zawsze te ścieżki, które mają najmniejszy koszt.

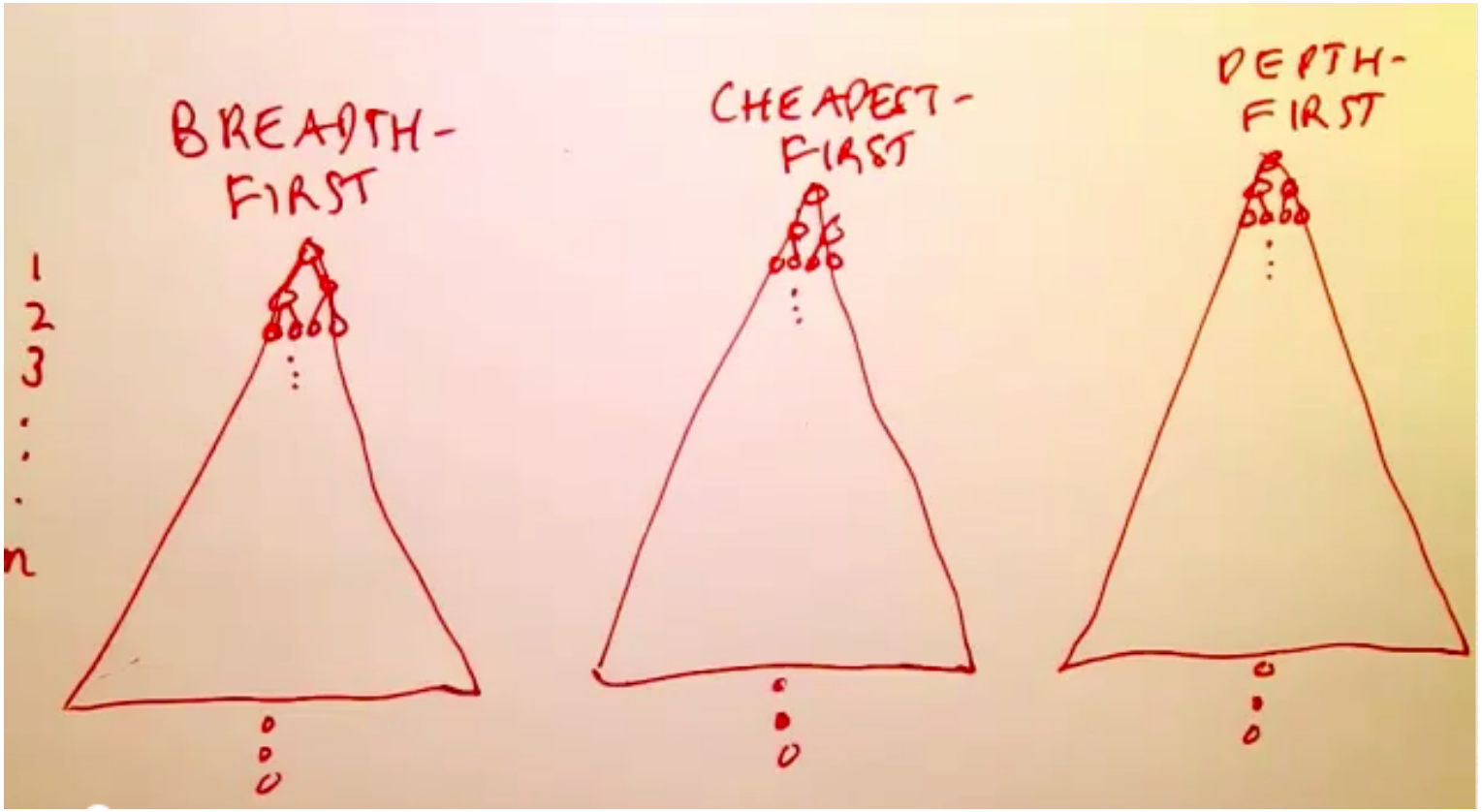


Porównanie trzech strategii



Przewaga przeszukiwania w głąb (mało pamięci)

Założenie: skończone/nieskończone drzewo do przeszukania.



Główne cechy algorytmu:

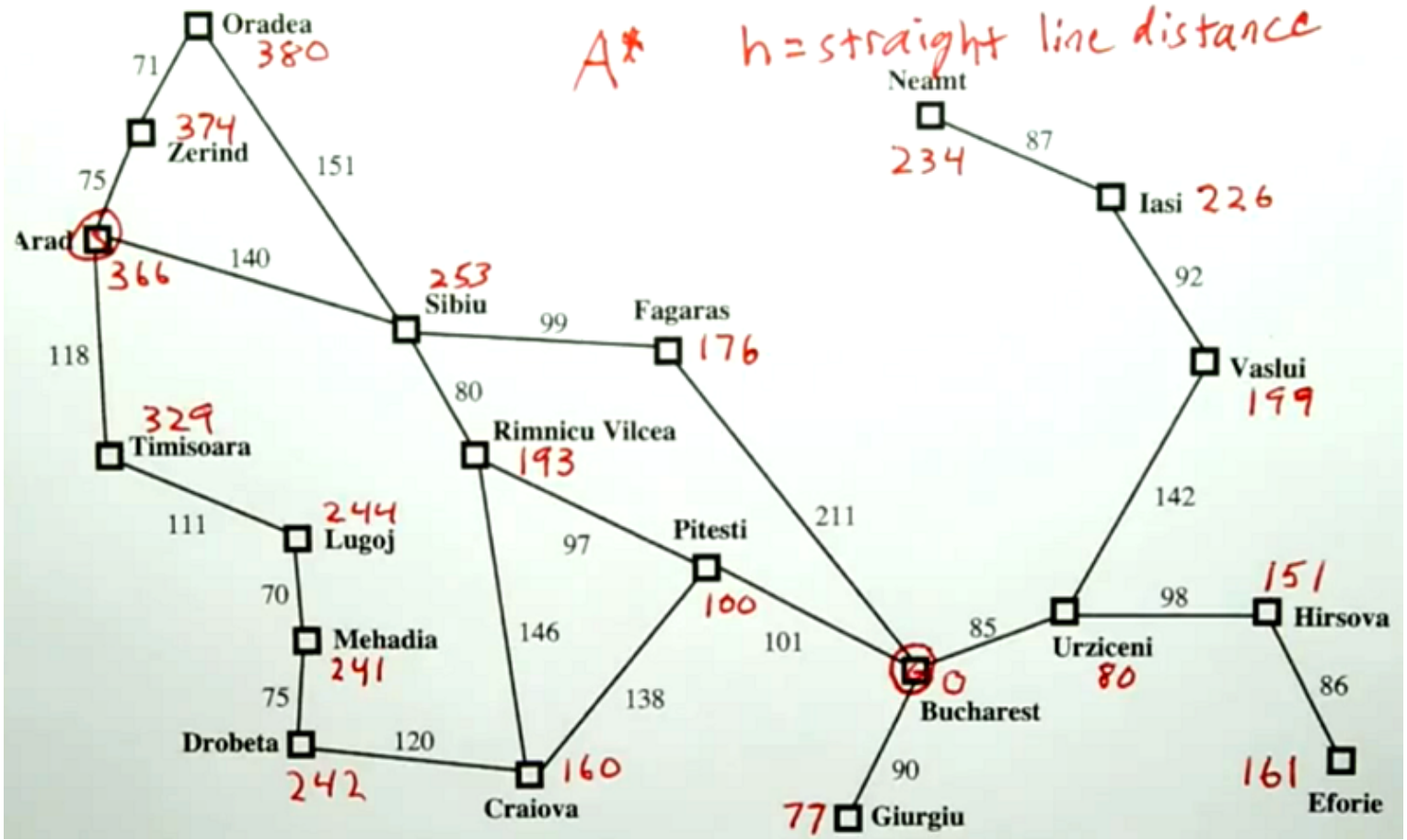
Funkcja oceny dla A*

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

gdzie:

- $g(n)$ = koszt osiągnięcia bieżącego węzła n z węzła początkowego, koszt ścieżki
- $h(n)$ = oszacowany koszt przejścia z węzła n do rozwiązania
- $f(n)$ = oszacowany pełny koszt ścieżki przez węzeł n do rozwiązania

A* - przykład

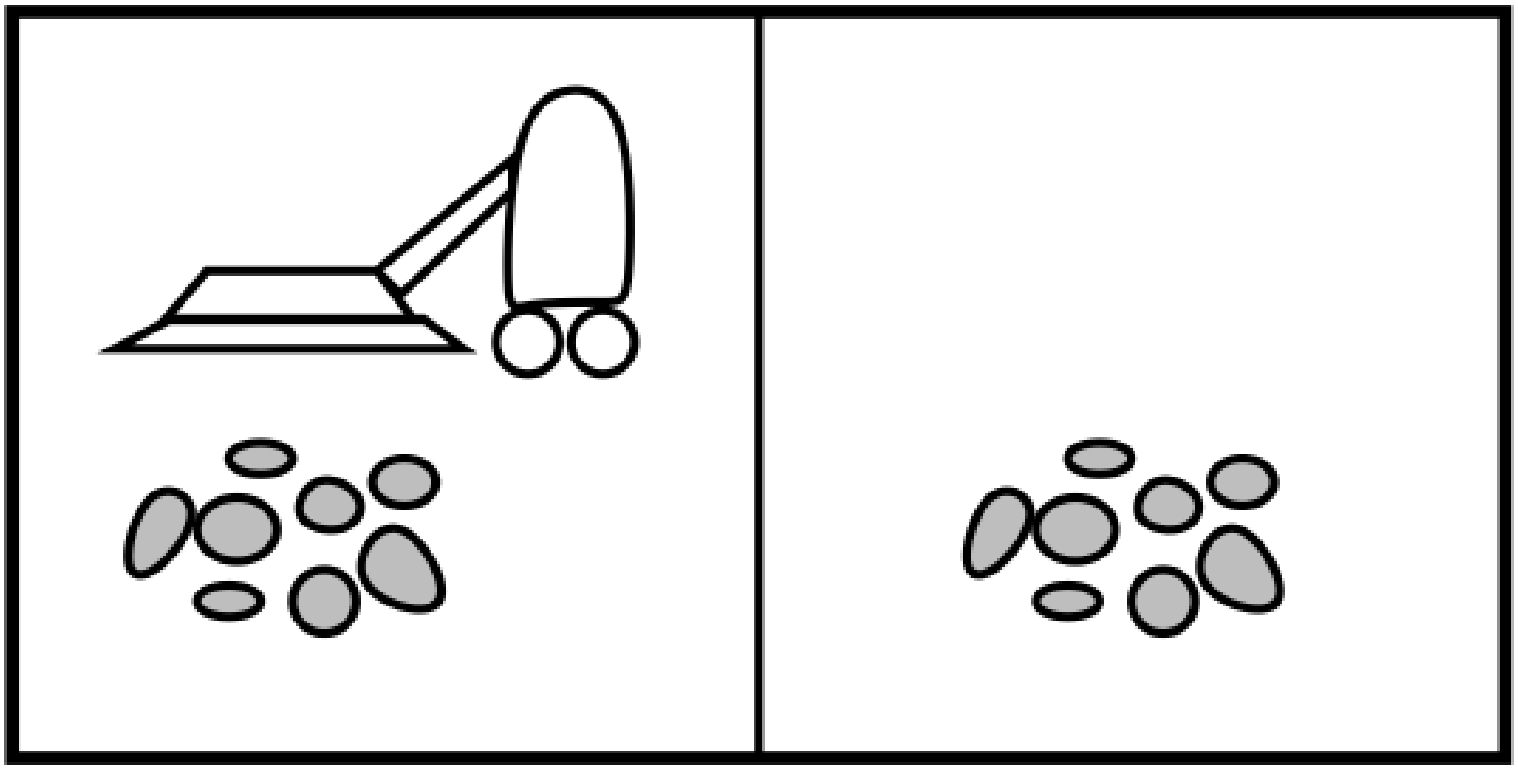

A* h=straight line distance
Oradea
380
71
374
Zerind
151
75
Arad
366
140
253
Sibiu
99
Fagaras
176
118
329
Timisoara
80
Rimnicu Vilcea
193
111
244
Lugoj
97
Pitesti
211
70
100
Mehadia
241
146
101
75
138
Drobeta
120
242
160
Craiova
Neamt
234
87
Iasi
226
92
Vaslui
199
142
151
98
Hirsova
85
Urziceni
80
0
Bucharest
86
90
161
Eforie
77
Giurgiu

A* - pytanie

Czy zawsze znajduje ścieżkę o najmniejszym koszcie?

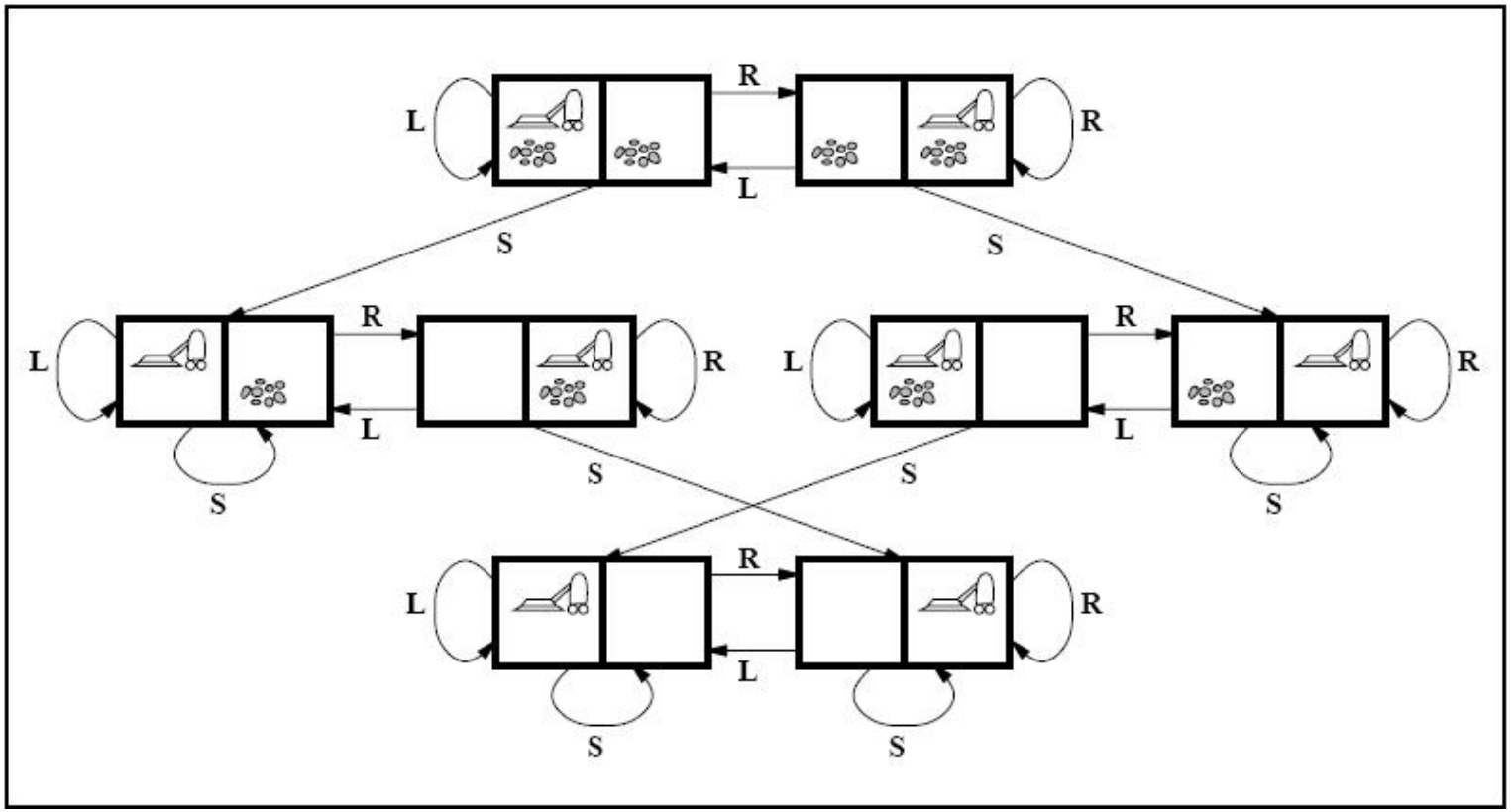
- tak, zawsze
- nie, zależy to od rozwiązywanego problemu
- nie, zależy od funkcji h

Przykład innej przestrzeni stanów

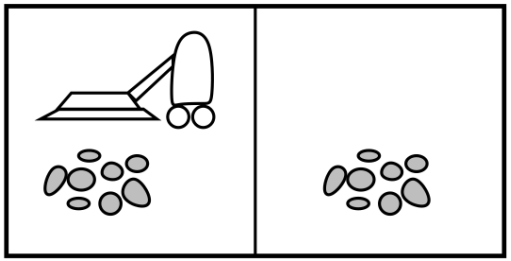


Z ilu stanów składa się ten uproszczony świat?

Przykład innej przestrzeni stanów



A co jeżeli problem się komplikuje?



- Robot ma włącznik, który ma trzy stany $\{on, off, sleep\}$
- Czujnik brudu, który ma dwa stany $\{on, off\}$
- Ustawienie szczotek na 5 pozycji $\{1, 2, 3, 4, 5\}$
- Zamiast dwóch pól jest 10.

Ile będzie stanów?

Kiedy przeszukiwanie/planowanie działa?

- Środowisko jest w pełni obserwowalne, musimy określić stan początkowy.
- Dziedzina musi być znana, by określić listę operatorów/akcji.
- Dziedzina musi być dyskretna
- Deterministyczna, musi być znany rezultat wykonania akcji.
- Statyczna, środowisko nie może się zmieniać.