SID – Wykład 1
Wprowadzenie

Dominik Ślęzak

Wydział Matematyki, Informatyki i Mechaniki UW
slezak@mimuw.edu.pl

Program przedmiotu

- algorytmy heurystyczne
- problemy optymalizacyjne
- strategie w grach
- wnioskowanie w logice
- planowanie
- systemy decyzyjne i uczące się
- sieci neuronowe
- zbiory rozmyte i przybliżone
- sieci bayesowskie...

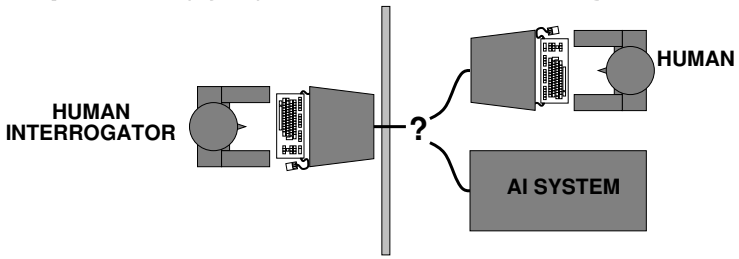
Literatura

- Stuart Russell, Peter Norvig
  Artificial Intelligence: A Modern Approach
  Prentice Hall 2003, wydanie II
  http://aima.cs.berkeley.edu
- George Luger
  Artificial Intelligence: Structures
  and Strategies for Complex Problem Solving
  Addison Wesley 2005, wydanie V
  http://www.cs.unm.edu/ luger/ai-final
- Tom Mitchell
  Machine Learning
  McGraw Hill 1997
  http://www-2.cs.cmu.edu/ tom/mlbook.html

Co to znaczy Sztuczna Inteligencja?

Turing (1950) “Computing machinery and intelligence”:

- “Czy maszyny mogą myśleć?” $\longrightarrow$ “Czy maszyny mogą zachowywać się inteligentnie?”
- Operacyjny test na inteligentne zachowanie:



Problem: test Turinga nie jest powtarzalny, konstruktywny, lub poddawalny matematycznej analizie

Co to znaczy Sztuczna Inteligencja?

Turing (1950) “Computing machinery and intelligence”:

- Zapowiadał, że przed rokiem 2000 maszyna będzie mieć 30% szans na udane imitowanie inteligencji człowieka przez 5 minut wobec przeciętnej osoby
- Przewidział wszystkie główne argumenty skierowane przeciwko sztucznej inteligencji w ciągu kolejnych 50 lat
- Zaproponował jako główne elementy SI: wiedzę, wnioskowanie, język, rozumienie, uczenie

Symulowanie ludzkiego myślenia

Lata 60-te “rewolucja kognitywna”: psychologia przetwarzania informacji zastąpiła dominującą koncepcję behawioryzmu

Wymaga naukowych teorii o wewnętrznym działaniu umysłu: Jaki poziom abstrakcji? “Wiedza” czy “układy”? Jak weryfikować? Wymaga:

1. Przewidywania i testowania zachowania ludzkiego podmiotu (top-down)
2. Bezpośredniego rozpoznawania na podstawie sygnałów neurologicznych (bottom-up)

Żadne z tych dwu podejść (Nauka Kognitywna oraz Nauka Neurokognitywna) nie jest Sztuczną Inteligencją, ale wszystkie trzy mają wspólna cechę:

- dotychczasowe teorie nie wyjaśniają niczego przypominającego typową ludzką inteligencję

Myślenie racjonalne

Normatywne (lub wyznaczone) raczej niż opisowe

Arystoteles: jakie są poprawne argumenty/procesy myślowe?

Kilka greckich szkół rozwinęło różne formy logiki:
notację i reguły wnioskowania dla myśli
mogły one stanowić poprzedzenie idei mechanizacji

Bezpośrednia linia prowadząca do współczesnej SI została wyznaczona przez matematykę i filozofię

Problemy:

1. Nie wszystkie inteligentne zachowania są związane z logicznym wnioskowaniem
2. Jaki jest cel myślenia? Jakie myśli powinienem mieć?

Prehistoria Sztucznej Inteligencji

Filozofia	logika, metody wnioskowania
	umysł jako fizyczny system
	podstawy uczenia, języka, racjonalności
Matematyka	formalna reprezentacja i dowód
	algorytmy, obliczenia,
	(nie-)rozstrzygalność, (nie-)konstruktywność
	prawdopodobieństwo
Psychologia	adaptacja
	zjawisko postrzegania i kontroli motorycznej
	techniki eksperymentalne (psychofizyka, etc.)
Ekonomia	formalna teoria podejmowania racjonalnych decyzji
Lingwistyka	reprezentacja wiedzy
	gramatyka
Neuronauka	podłoże fizyczne aktywności umysłowej
Teoria sterowania	systemy homeostatyczne, stabilność
	proste projekty optymalnych agentów

Ścisła historia Sztucznej Inteligencji

1943	McCulloch & Pitts: Model mózgu jako układ boolowski
1950	Artykuł Turinga “Computing Machinery and Intelligence”
1952–69	Okres rozkwitu:
1950s	Wczesne programy SI, w tym program grający w warcaby Samuela, Logic Theorist Newella i Simona, Geometry Engine Gelertner’a
1956	Spotkanie w Dartmouth: powstaje termin “Sztuczna Inteligencja”
1965	Pełna metoda rezolucji Robinsona do wnioskowania w logice I rzędu
1966–74	Złożoność obliczeniowa, badania sieci neuronowych zanikają
1969–79	Wczesny rozwój systemów opartych na wiedzy
1980–88	Przemysłowy boom systemów doradczych
1988–93	Przemysł systemów doradczych przeżywa regresję: “Zima SI”
1985–95	Sieci neuronowe wracają do popularności
1988–	rozwój badań związanych z prawdopodobieństwem ogólny wzrost poziomu zaawansowania technicznego systemów “nowości SI”: sztuczne życie, algorytmy genetyczne, soft computing
1995–	systemy wieloagentowe . . .

Co SI potrafi dzisiaj

- Rozegrać przyzwoity mecz tenisa stołowego
- Prowadzić samochód po krętej, górskiej drodze
- Prowadzić samochód w centrum Kairu
- Zrobić zakupy spożywcze na tydzień w supermarkecie Berkeley Bowl
- Zrobić zakupy spożywcze na tydzień w internecie
- Rozegrać przyzwoitą partię brydża
- Odkryć i udowodnić nowe twierdzenie matematyczne
- Wymyślić zabawną historię
- Udzielić kompetentnej porady prawnej w wyspecjalizowanym zakresie prawa
- Tłumaczyć mówiony angielski na mówiony szwedzki w czasie rzeczywistym
- Wykonać skomplikowaną operację chirurgiczną

Modelowanie problemów praktycznych

Reprezentacja problemu:

- stany: reprezentują opisy różnych stanów świata rzeczywistego
- akcje: reprezentują działania zmieniające bieżący stan
- koszt akcji (≥ 0): reprezentuje koszt związany z wykonaniem akcji

Świat rzeczywisty jest ogromnie złożony: przestrzeń stanów dla problemu musi być wyabstrahowana z rzeczywistości, pojedyncze akcje w opisie problemu muszą reprezentować złożone operacje rzeczywiste

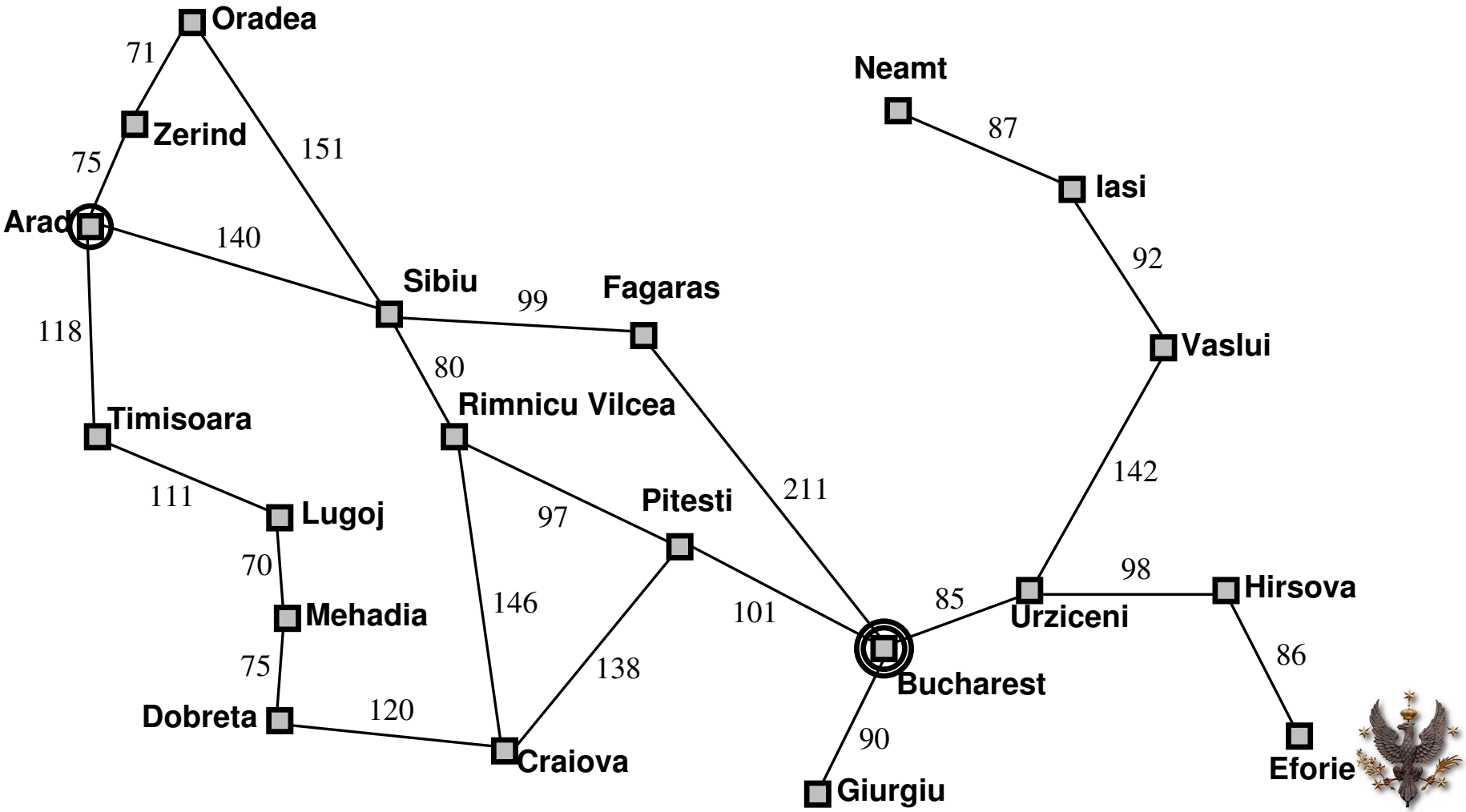
Modelowanie problemów praktycznych

Sformułowanie problemu:

- stan początkowy: początkowy stan przed rozwiązaniem problemu
- cel: stan docelowy lub formuła oceniająca, czy dany stan spełnia cel
- rozwiązanie: ciąg akcji prowadzący od stanu początkowego do celu
- koszt rozwiązania: funkcja oceny kosztu rozwiązania równa sumie kosztów poszczególnych akcji występujących w rozwiązaniu

Rozwiązania o niższym koszcie są lepsze niż rozwiązania o wyższym koszcie.

Przykład problemu: Rumunia



Przykład problemu: Rumunia

Obecnie w Aradzie. Samolot odlatuje jutro z Bukaresztu

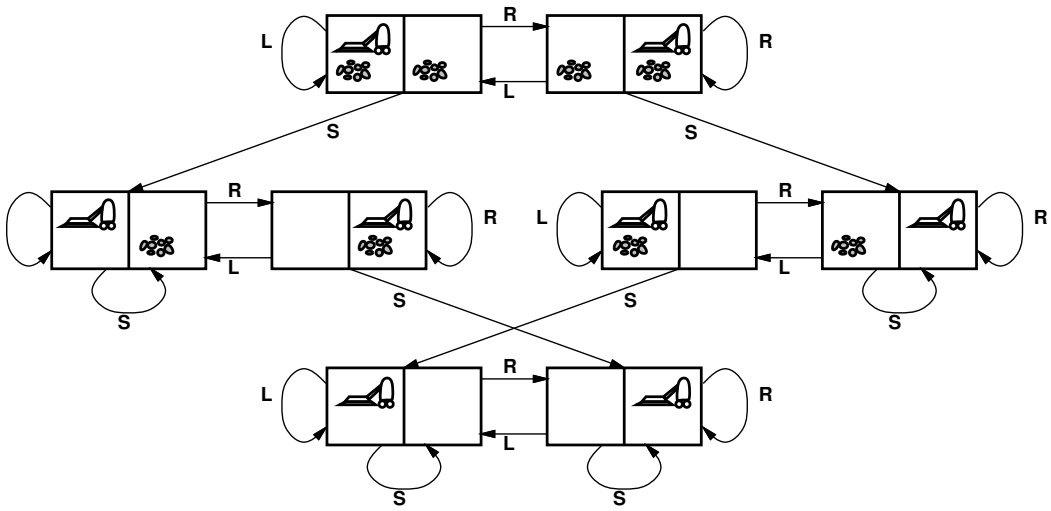
Reprezentacja problemu:

- stany: miasta {Arad, Sibiu, Tibisoara, Zerind, ...}
- akcje: przejazdy pomiędzy dwoma miastami, np. Arad $\rightarrow$ Zerind
- koszt akcji: odległość pomiędzy dwoma miastami

Sformułowanie problemu:

- stan początkowy: Arad
- stan docelowy: Bukareszt
- rozwiązanie: ciąg przejazdów, np. Arad $\rightarrow$ Sibiu $\rightarrow$ Fagaras $\rightarrow$ Bukareszt
- koszt rozwiązania: suma kilometrów pomiędzy kolejnymi miastami

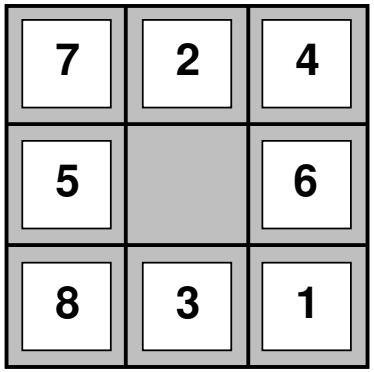
Przykład problemu: sprzątanie



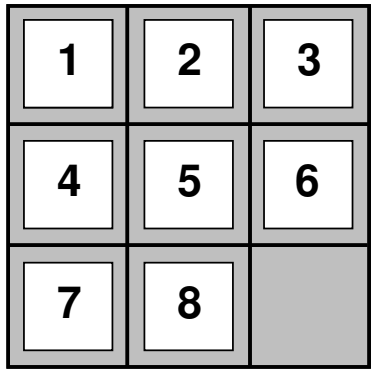
- stany: stan pomieszczeń (czysto/brudno) i lokalizacja robota
- akcje: *Lewo*, *Prawo*, *Odkurzaj*, *NicNieRob*
- cel: czysto
- koszt rozwiązania: 1 dla każdej akcji (0 dla *nicNieRob*)

Przykład problemu: 8-elementowe puzzle

- stany: rozmieszczenia puzzli
- akcje: przesuń puste miejsce w prawo, w lewo, w górę, w dół
- cel: wybrane rozmieszczenie puzzli
- koszt rozwiązania: 1 za każdy ruch



Start State

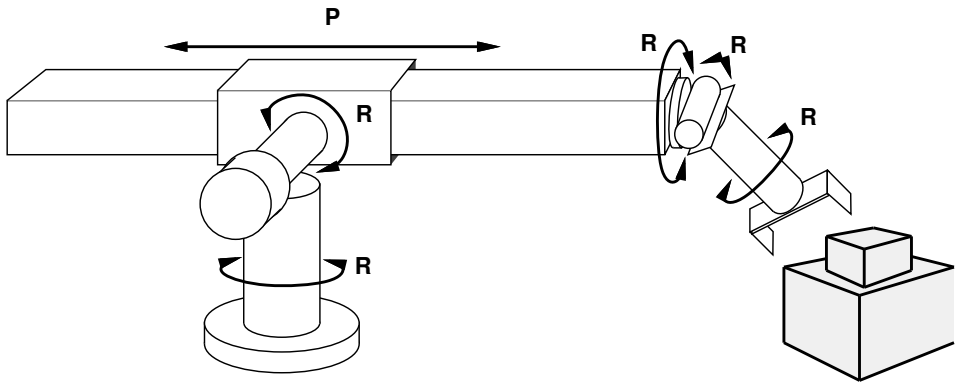


Goal State

Uwaga:

znalezienie optymalnego rozwiązania dla rodziny problemów *n*-elementowych puzzli jest NP-trudne

Przykład problemu: montaż przy użyciu robota



- stany: rzeczywiste współrzędne kątów w złączeniach robota, elementy do zmontowania
- akcje: ciągłe ruchy złączy robota
- cel: kompletny montaż
- koszt rozwiązania: czas montażu

Przykład problemu: Maximum Satisfiability problem (MAX-SAT)

$$(p_{11} \vee p_{12} \vee \ldots \vee p_{1k_1}) \wedge (p_{21} \vee p_{22} \vee \ldots \vee p_{2k_2}) \wedge \ldots \wedge (p_{n1} \vee p_{n2} \vee \ldots \vee p_{nk_n})$$

gdzie p_{ij} są literałami.

- stany: wartościowania zmiennych zdaniowych
- cel: określenie maksymalnej liczby klauzul danej formuły logicznej, które mogą być spełnione dla dowolnie wybranego wartościowania zmiennych zdaniowych
- koszt rozwiązania: liczba klauzul niespełnionych

Przeszukiwanie drzewa stanów

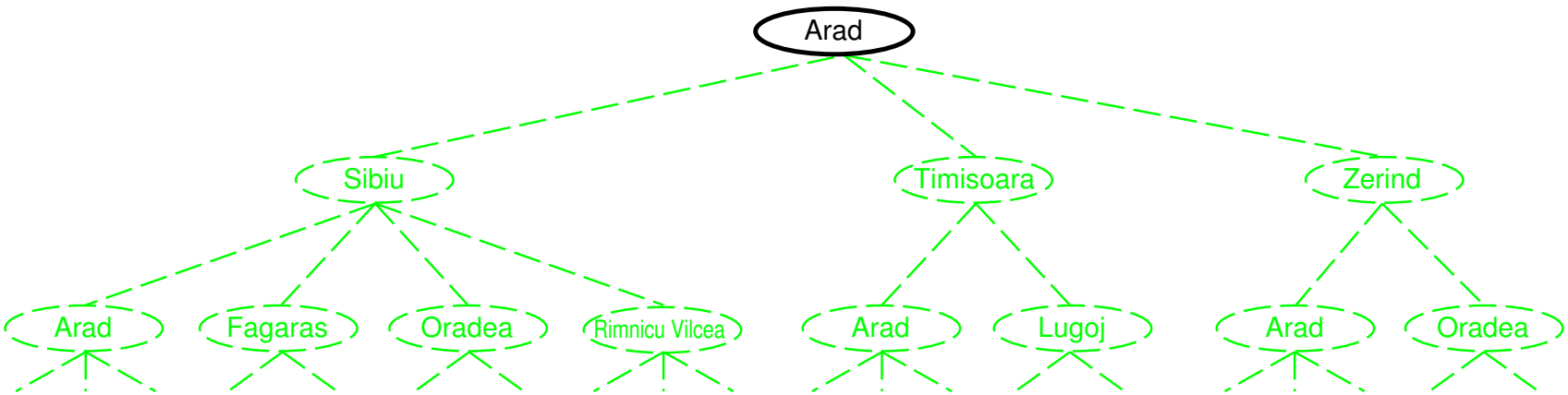
Prosty pomysł

Symulowanie offline przeszukiwania przestrzeni stanów poprzez generowanie następników wcześniej odwiedzonych stanów (znane również jako ekspansja stanów)

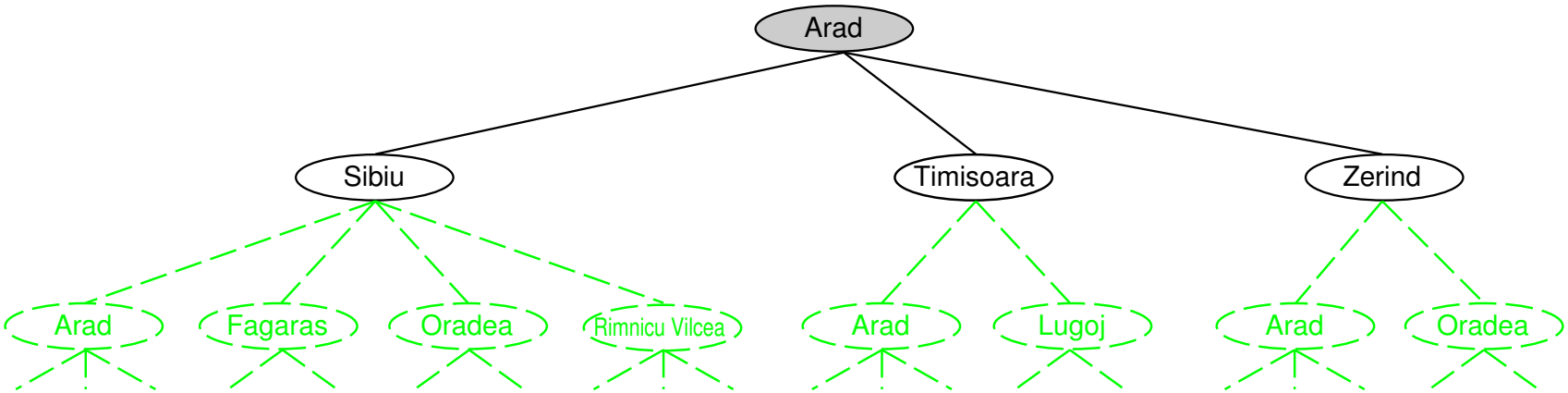
Przeszukiwanie drzewa stanów

function TREE-SEARCH(*problem*, *strategy*) **returns** a solution, or failure
    initialize the search tree using the initial state of *problem*
    loop
        if there are no candidates for expansion **then**
            return failure
        end if
        choose a leaf node for expansion according to *strategy*
        if the node contains a goal state **then**
            return the corresponding solution
        else
            expand the node and add the resulting nodes to the search tree
        end if
    end loop
end function

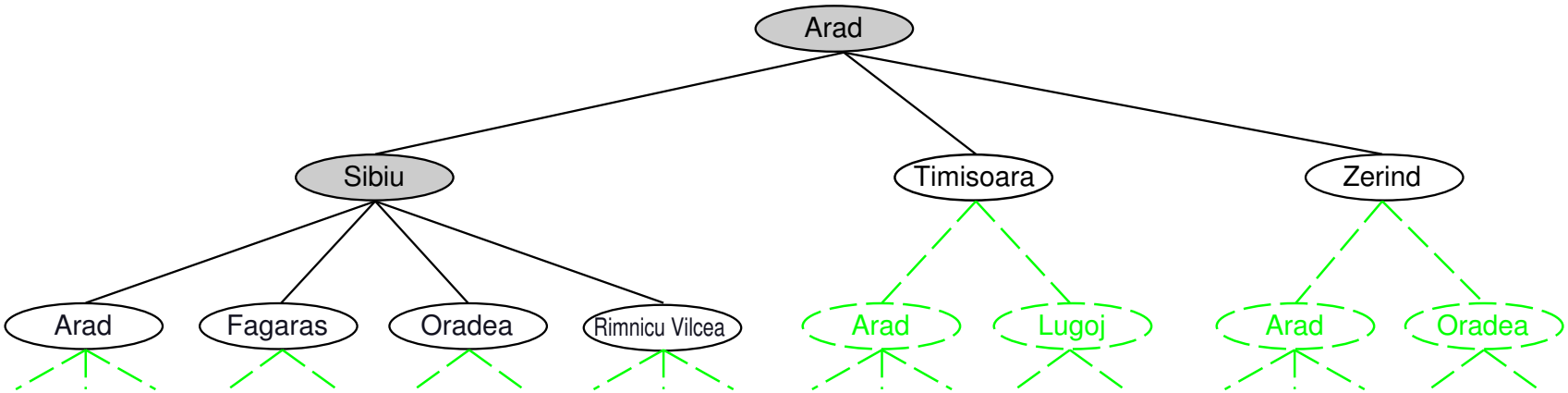
Przykład przeszukiwania drzewa stanów



Przykład przeszukiwania drzewa stanów

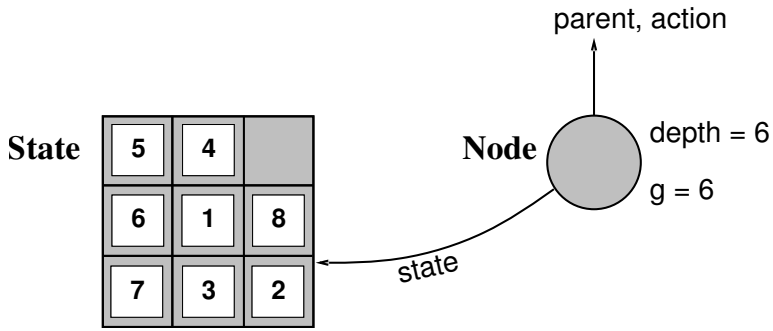


Przykład przeszukiwania drzewa stanów



Implementacja: stany vs. węzły

- Stan jest fizyczną konfiguracją (jej reprezentacją)
- Węzeł jest strukturą danych stanowiącą część drzewa przeszukiwań — zawiera poprzednik (*parent*), następniki, głębokość (*depth*) oraz koszt ścieżki od korzenia (*g*)
- Stany nie mają poprzedników, następników, głębokości i kosztu ścieżki!



Funkcja SuccessorFn zwraca jako wynik zbiór akcji możliwych do wykonania w danym stanie wraz ze stanami osiąganymi po wykonaniu akcji.

Funkcja Expand tworzy nowe węzły i wypełnia ich pola używając funkcji.

Implementacja: przeszukiwanie drzewa stanów

```
function TREE-SEARCH(problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop
        if fringe is empty then
            return failure
        end if
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds then
            return node
        end if
        fringe ← INSERT-ALL(EXPAND(node, problem), fringe)
    end loop
end function
```

```
function EXPAND(node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node; ACTION[s] ← action; STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    end for
end function
```

Strategie przeszukiwania

Strategia jest definiowana poprzez wybór kolejności ekspansji stanów.
Strategie są ocenianie według następujących kryteriów:

- zupełność — czy zawsze znajduje rozwiązanie, jeśli ono istnieje?
- złożoność czasowa — liczba wygenerowanych węzłów
- złożoność pamięciowa — maksymalna liczba węzłów w pamięci
- optymalność — czy znajduje rozwiązanie o minimalnym koszcie?

Złożoność czasowa i pamięciowa są mierzone w terminach:

- b — maksymalnego rozgałęzienia drzewa przeszukiwań
- d — głębokość rozwiązania o najmniejszym koszcie
- m — maksymalnej głębokości drzewa przeszukiwań (może być ∞)

Rodzaje strategii przeszukiwania

Strategie ślepe korzystają z informacji dostępnej jedynie w definicji problemu:

- Przeszukiwanie wszerz
- Strategia jednolitego kosztu
- Przeszukiwanie wgłąb
- Przeszukiwanie ograniczone wgłąb
- Przeszukiwanie iteracyjnie pogłębiane
- Przeszukiwanie dwukierunkowe

Rodzaje strategii przeszukiwania

Strategie heurystyczne korzystają z dodatkowej, heurystycznej funkcji oceny stanu, np. szacującej koszt rozwiązania od bieżącego stanu do celu:

- Przeszukiwanie zachłanne
- Przeszukiwanie A*
- Rekurencyjne przeszukiwanie pierwszy najlepszy
- Przeszukiwanie lokalne zachłanne (hill-climbing)
- Symulowane wyżarzanie
- Algorytm genetyczny

Przeszukiwanie wszerz

Wykonuje ekspansję najpłytszego węzła spośród tych, które nie były jeszcze rozszerzone (implementacja: *fringe* jest kolejką FIFO, tzn. nowe następniki dodawane są na koniec kolejki).

```
function TREE-SEARCH(problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop
        if fringe is empty then
            return failure
        end if
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem] applied to STATE(node) succeeds then
            return node
        end if
        fringe ← INSERT-ALL(EXPAND(node, problem), fringe)
    end loop
end function
```

Strategia jednolitego kosztu

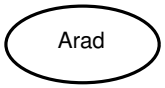
Wykonuje ekspansję węzła o najmniejszym koszcie spośród tych, które nie były jeszcze rozszerzone.

Implementacja

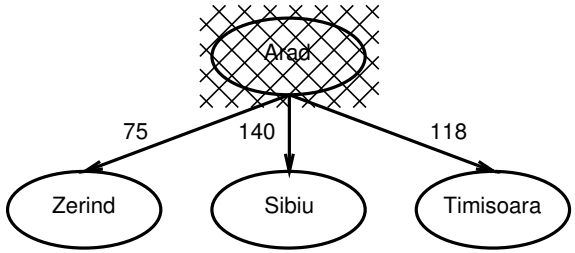
fringe jest kolejką priorytetowa porządkującą węzły według kosztu ścieżki od korzenia

Odpowiada przeszukiwaniu wszerz jeśli koszt wszystkich pojedynczych akcji jest ten sam.

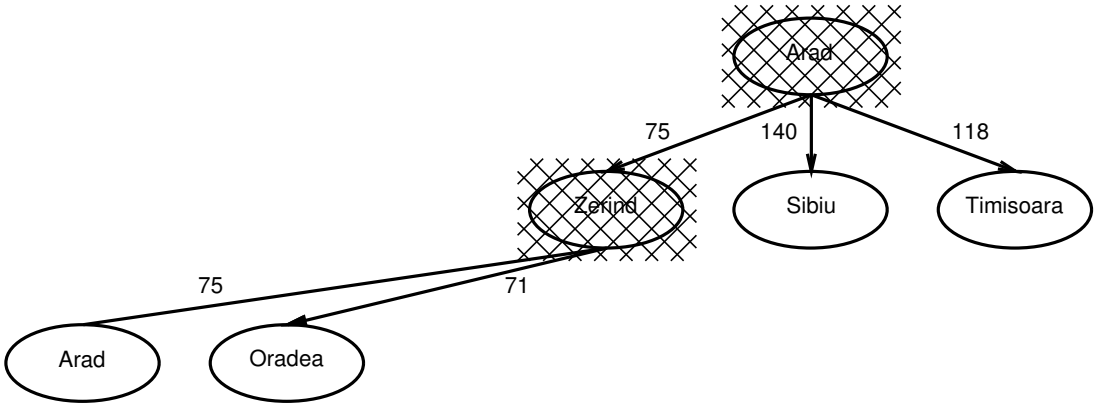
Strategia jednolitego kosztu



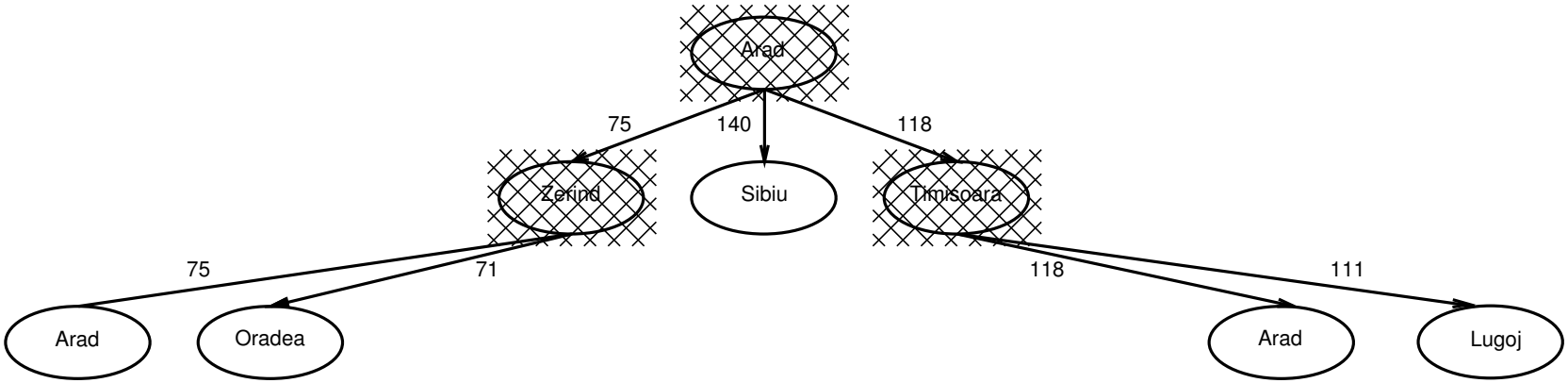
Strategia jednolitego kosztu



Strategia jednolitego kosztu



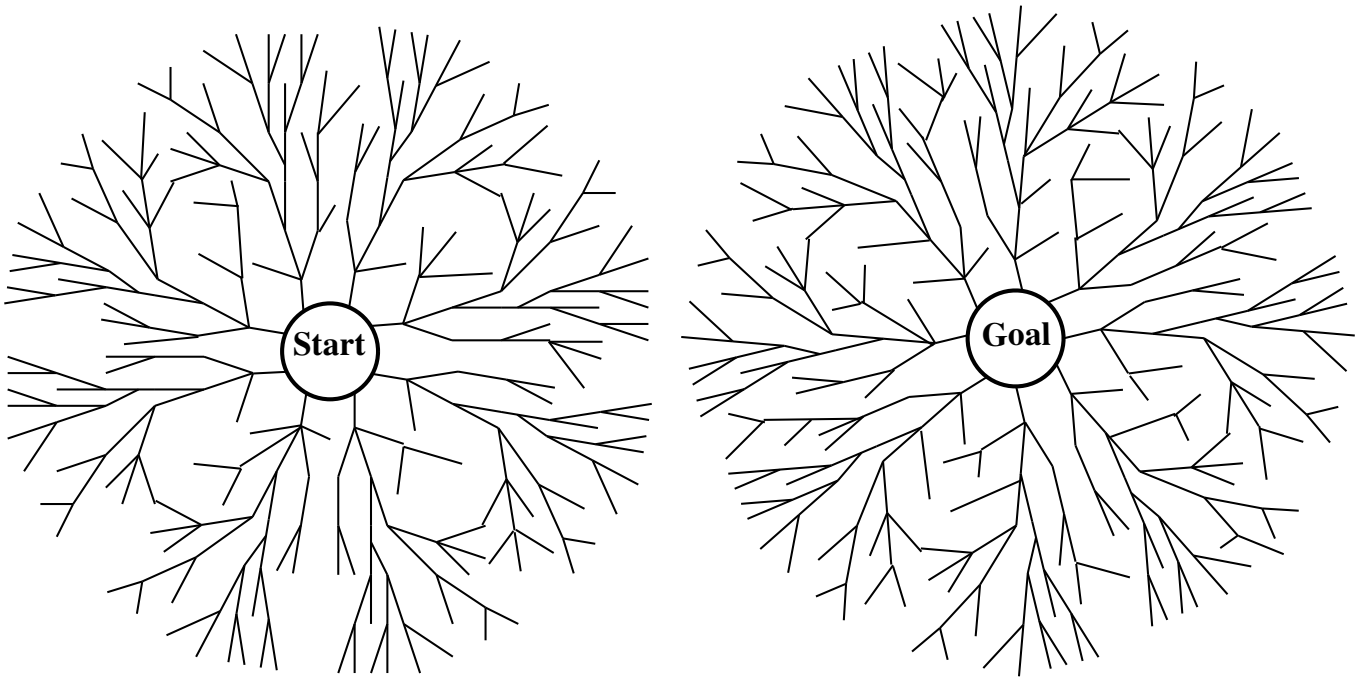
Strategia jednolitego kosztu



Przeszukiwanie dwukierunkowe

Wykonuje równolegle dwa przeszukiwania:

1. przeszukiwanie wprzód od stanu początkowego
2. przeszukiwanie w tył od stanu końcowego



Dziękuję za uwagę!