Artificial & Computational Intelligence

AIML CLZG557

M1 : Introduction
&
M2 : Problem Solving Agent using Search

Raja vadhana P

Assistant Professor,

BITS - CSIS

BITS Pilani

Pilani Campus

Course Plan

M1	Introduction to AI
M2	Problem Solving Agent using Search
M3	Game Playing
M4	Knowledge Representation using Logics
M5	Probabilistic Representation and Reasoning
M6	Reasoning over time
M7	Ethics in AI

Learning Objective

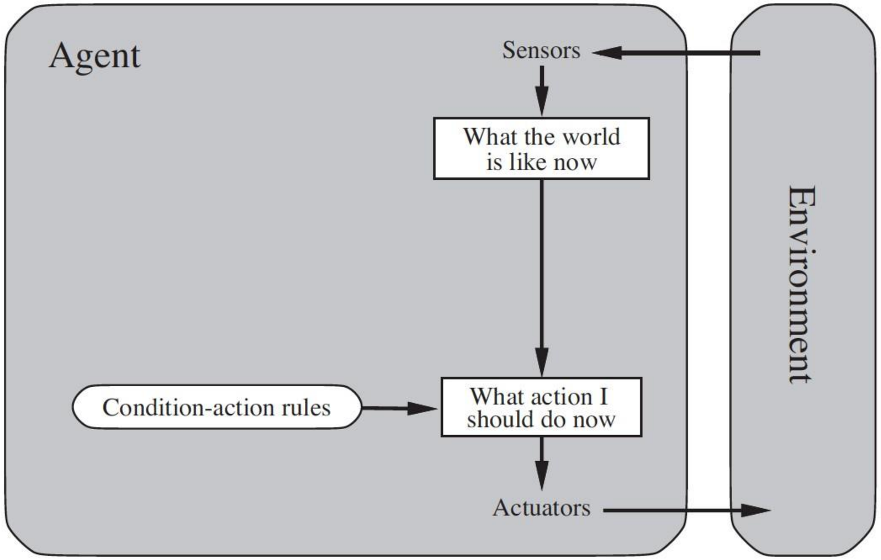
At the end of this class , students Should be able to:

1. Design problem solving agents
2. Create search tree for given problem
3. Apply uninformed search algorithms to the given problem
4. Compare performance of given algorithms in terms of completeness, optimality, time and space complexity
5. Differentiate for which scenario appropriate uninformed search technique is suitable and justify

Agents Architectures

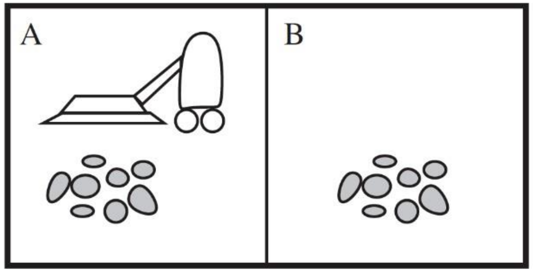
Reflex Agent

function SIMPLE-REFLEX-AGENT(*percept*) returns an action
    persistent: *rules*, a set of condition–action rules
    state←INTERPRET-INPUT(*percept*)
    rule←RULE-MATCH(*state, rules*)
    action ←*rule*.ACTION
    return *action*



function REFLEX-VACUUM-AGENT([*location*,*status*]) **returns** an action
    if *status* = *Dirty* **then return** Suck
    else if *location* = *A* **then return** Right
    else if *location* = *B* **then return** Left

Simple Reflex Agents

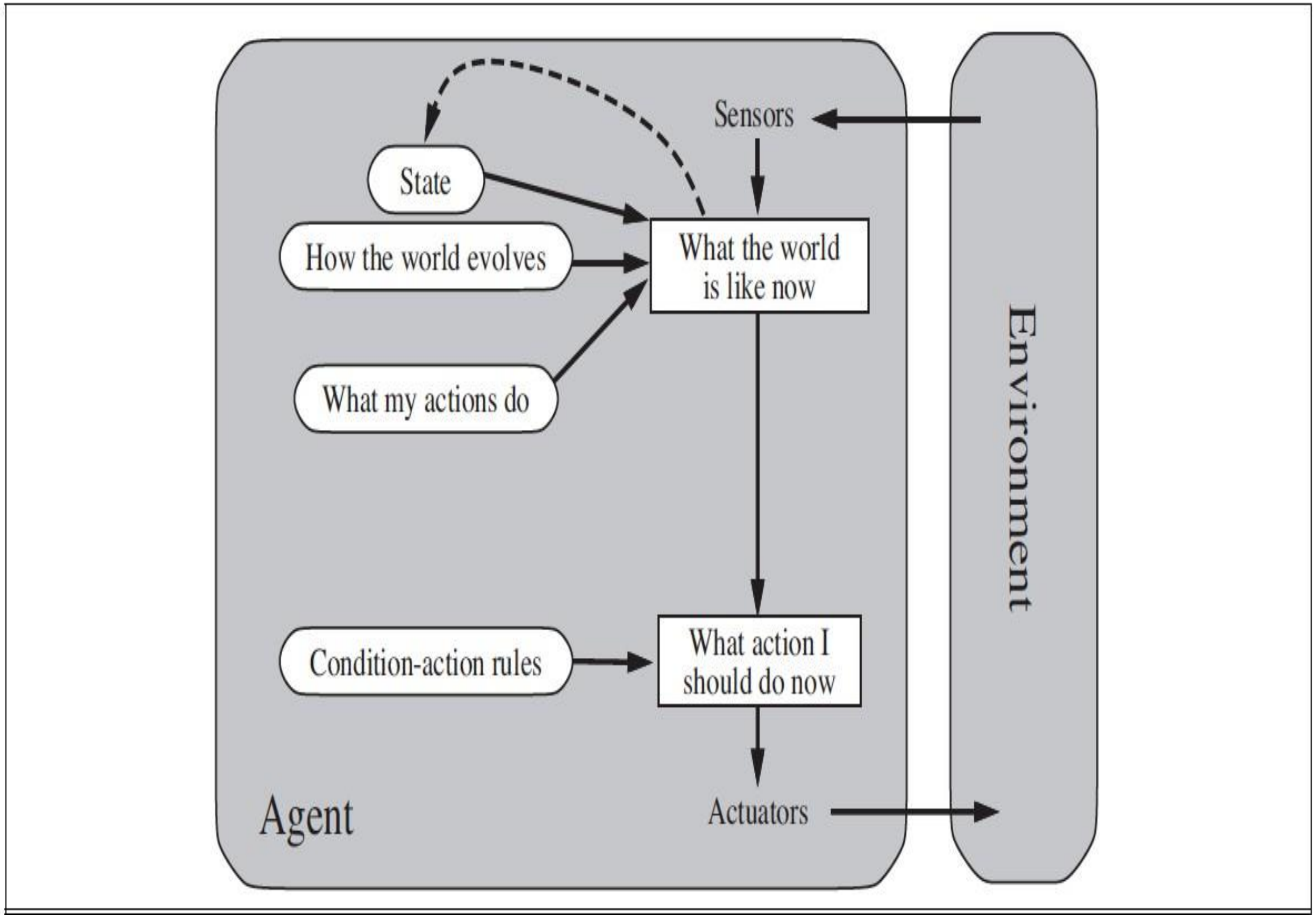


Agent Architectures

Model based Agent

Simple Reflex Agents

Model Based Agents



Model based Agent

function MODEL-BASED-REFLEX-AGENT(*percept*) **returns** an action

persistent: *state*, the agent's current conception of the world state
transition model, a description of how the next state depends on the current state and action
sensor model , a description of how the current world state is reflected in the agent's percepts
rules, a set of condition–action rules
action, the most recent action, initially none

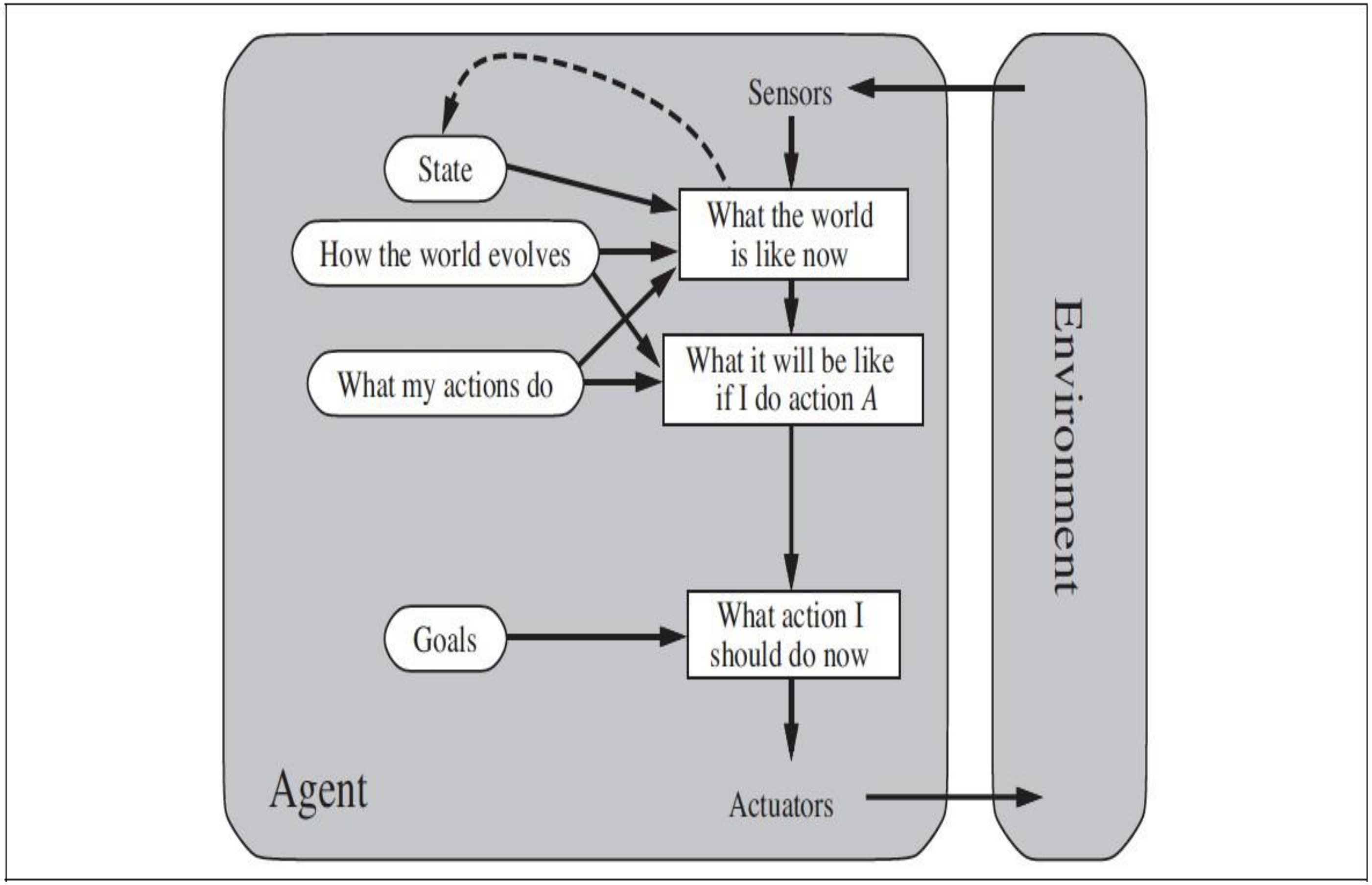
state←UPDATE-STATE(*state, action, percept, transition model, sensor model*)
rule←RULE-MATCH(*state, rules*)
action ←*rule*.ACTION
return *action*

Agent Architectures

Simple Reflex Agents

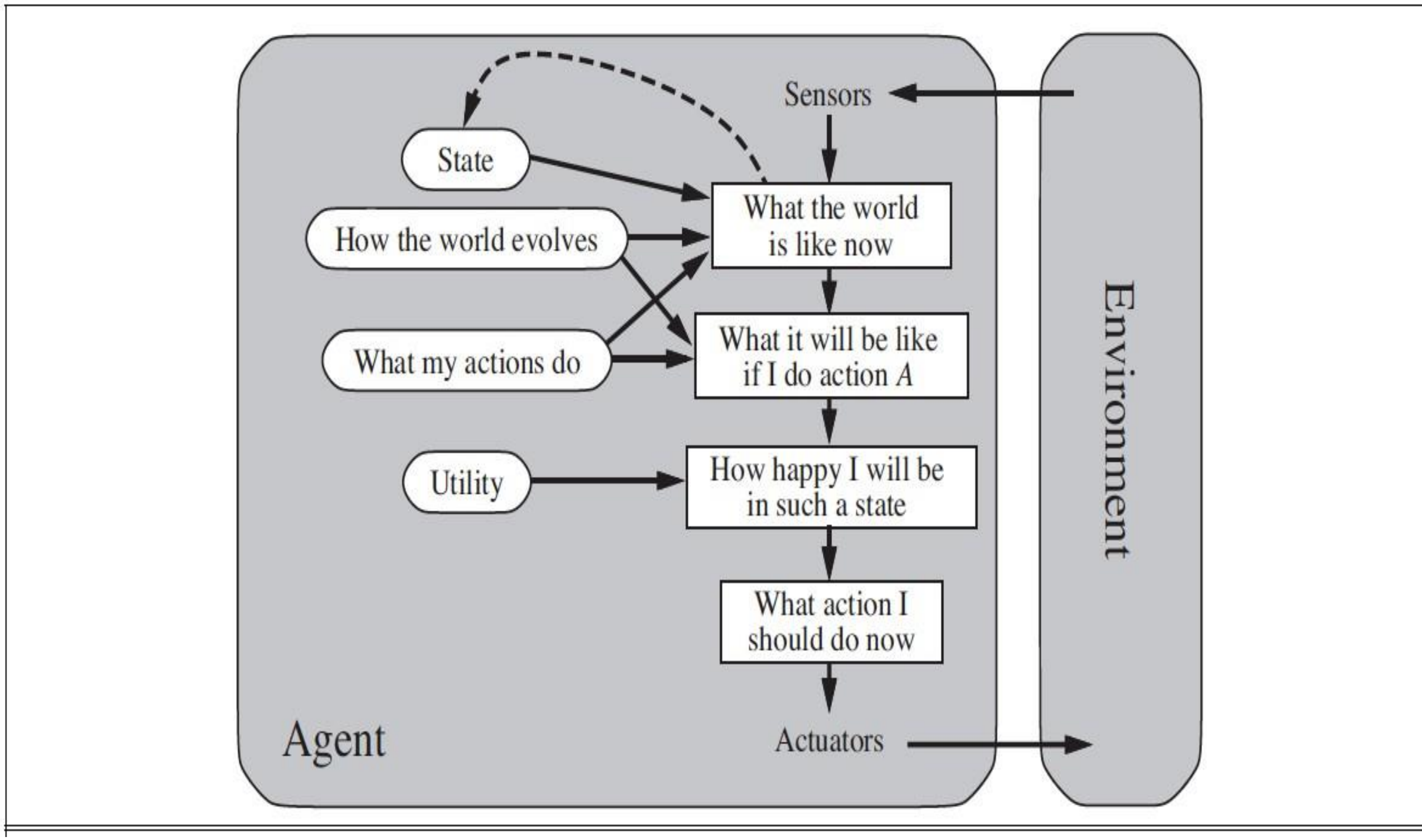
Model Based Agents

Goal Based Agents



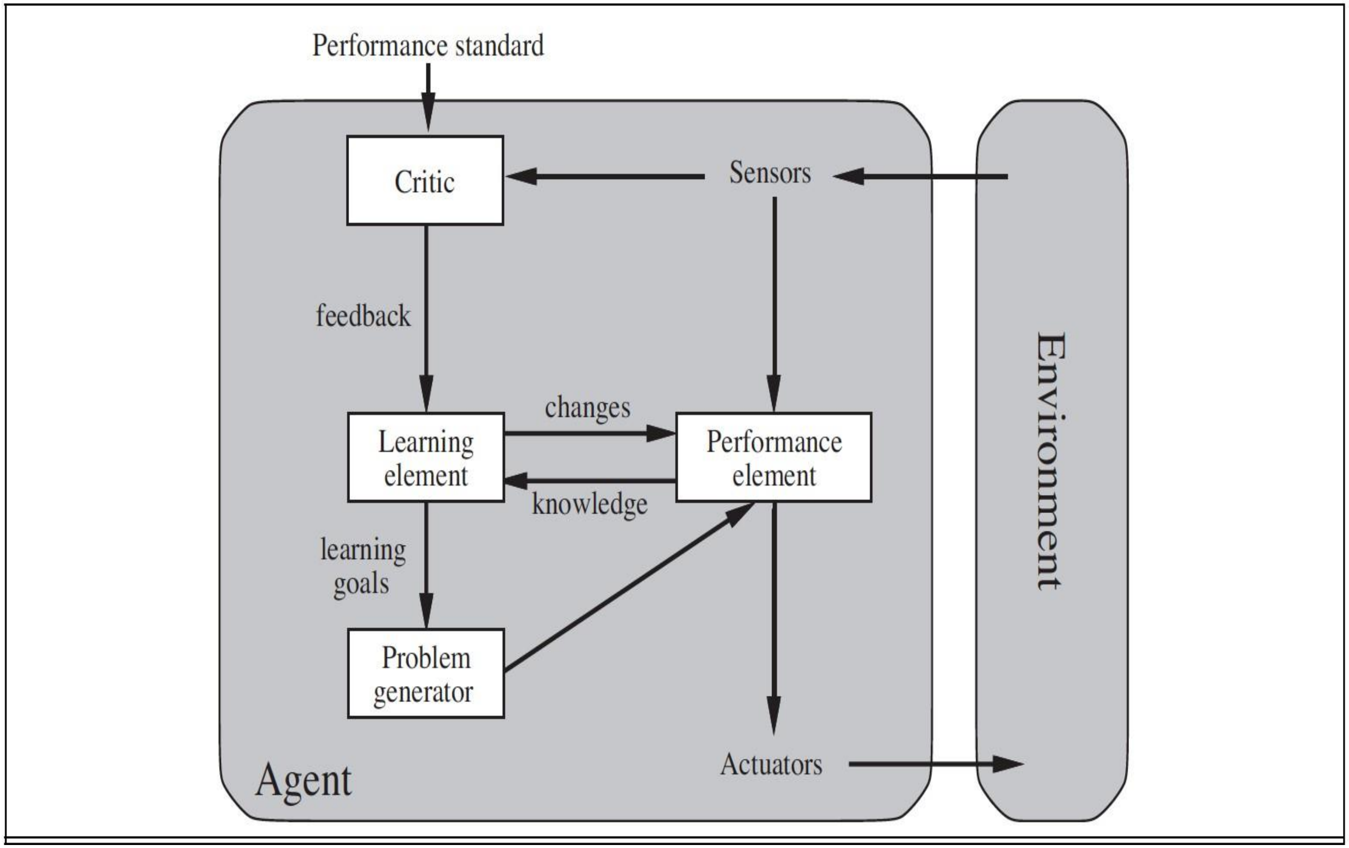
Agent Architectures

- Simple Reflex Agents
- Model Based Agents
- Goal Based Agents
- Utility Based Agents



Agent Architectures

- Simple Reflex Agents
- Model Based Agents
- Goal Based Agents
- Utility Based Agents
- Learning Agents



Role of Learning

Agents that improve their performance by learning from their own experiences

Input Percept

Output Action

Performance Element

APPLY BRAKE

Resulting State

Feedback

Critic

-2

Role of Learning

Input Percept

Possible Actions

Selected Action

Brake
Change Gear to Lower
Change Gear to Higher
Accelerate
Steer left
Steer right

Random

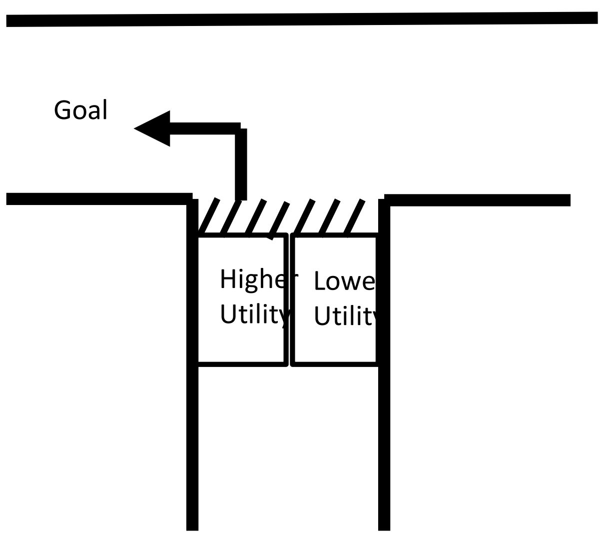
Change Gear to Lower

Role of Learning

Performance Element – Takes decision on action based on percept

$$f(red\ signal, \quad distance) = 15k\ N\ brake$$
$$distance = f'(percept\ sequence)$$
$$f(percepts, distance, raining)$$

- $f(state_0, actionA) = 0.83,$
- $f(state_0, actionB) = 0.45$

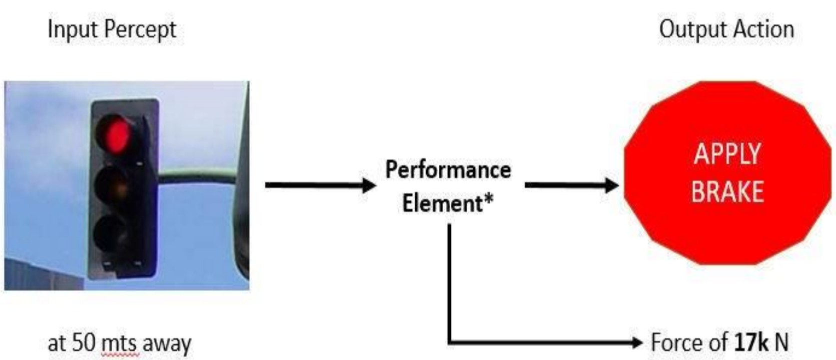


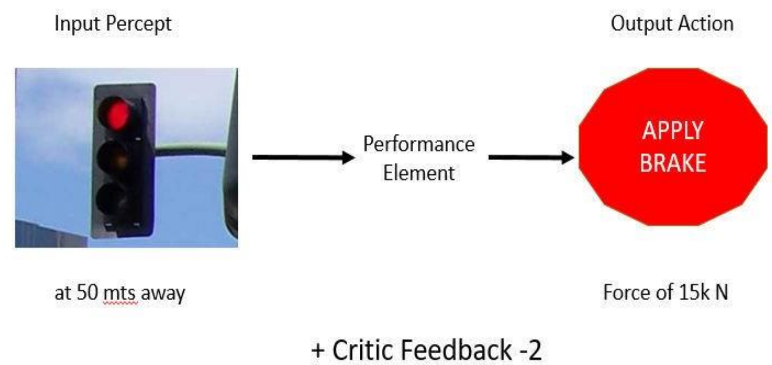
Role of Learning

Critic – Provides feedback on the actions taken

Learning :

Supervised Vs Unsupervised Vs Reinforcement

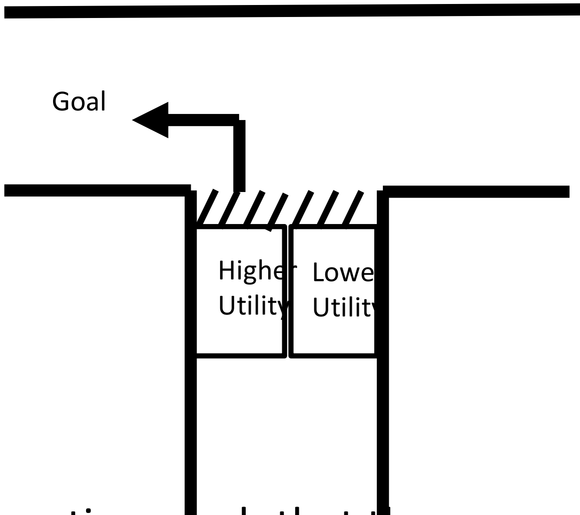




Role of Learning

Performance Element – Takes decision on action based on percept

$$f(red\ signal, \quad distance) = 15k\ N\ brake$$
$$distance = f'(percept\ sequence)$$
$$f(percepts, distance, raining)$$



- $f(state_0, actionA) = 0.83,$
- $f(state_0, actionB) = 0.45$

Learning Element – Make the performance element select better actions such that the utility function is optimized

Critic – Provides feedback on the actions taken

Problem Generator – Make the Performance Element select sub-optimal actions such that you would learn from unseen actions

Problem Formulation

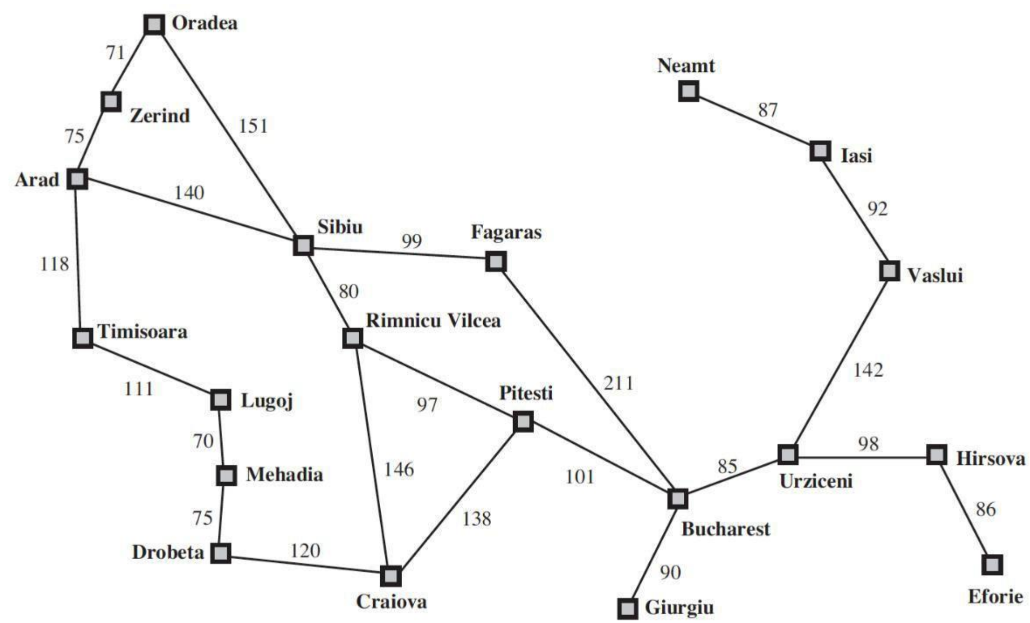
Problem Solving Agents

Goal based decision making agents finds sequence of actions that leads to the desirable state.

Phases of Solution Search by PSA

Goal Formulation → Problem Formulation → Search Phase → Execution Phase

Optimizes the Objective (Local | Global) Limits the Actions



Problem Solving Agents

Goal Formulation

Problem Formulation

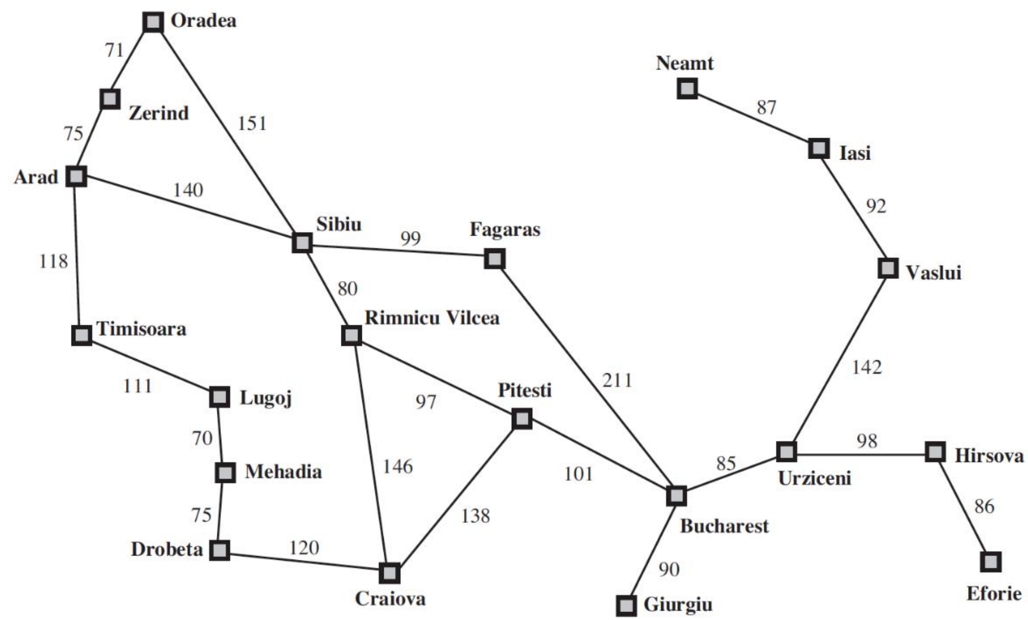
Search Phase

Execution Phase

Phases of Solution Search by PSA

State Space Creations [in the path of Goal]

Lists the Actions



Problem Solving Agents

Goal Formulation

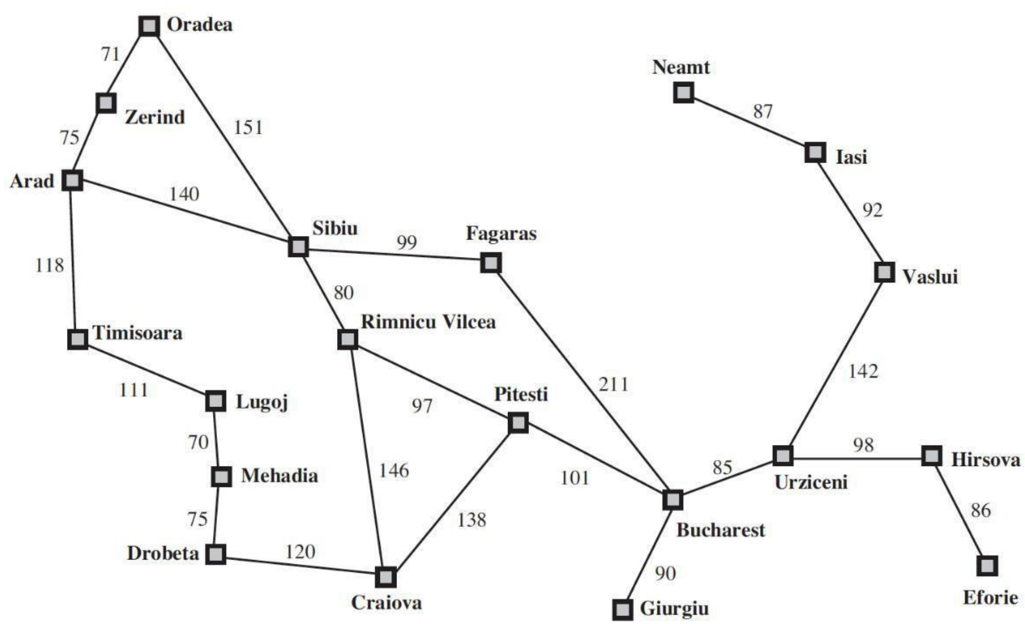
Problem Formulation

Search Phase

Execution Phase

Phases of Solution Search by PSA

Assumptions – Environment :
Static
Observable Discrete
Deterministic



Problem Solving Agents

Phases of Solution Search

Goal Formulation

↓

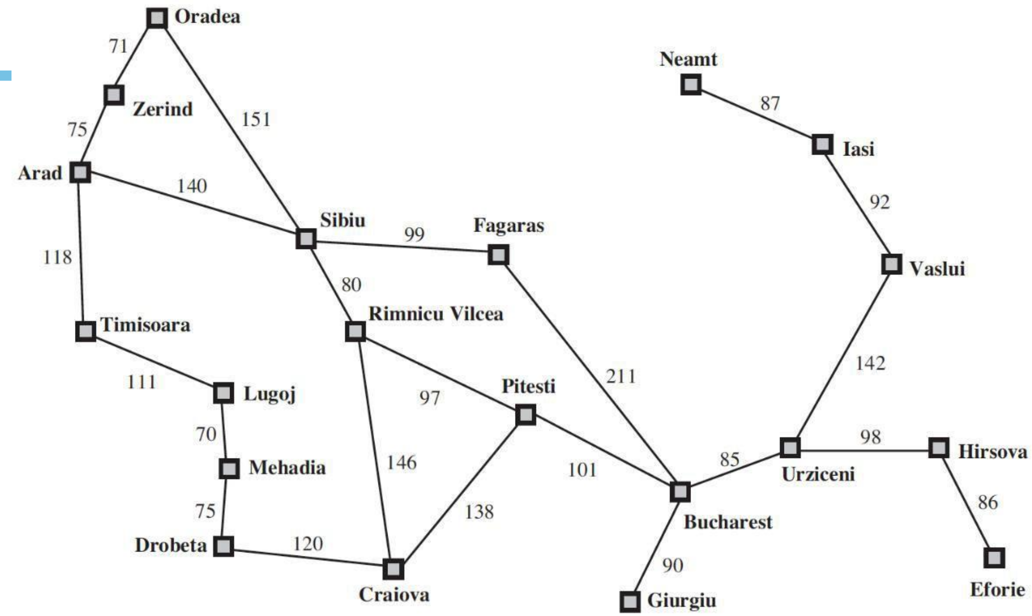
Problem Formulation

↓

Search Phase

↓

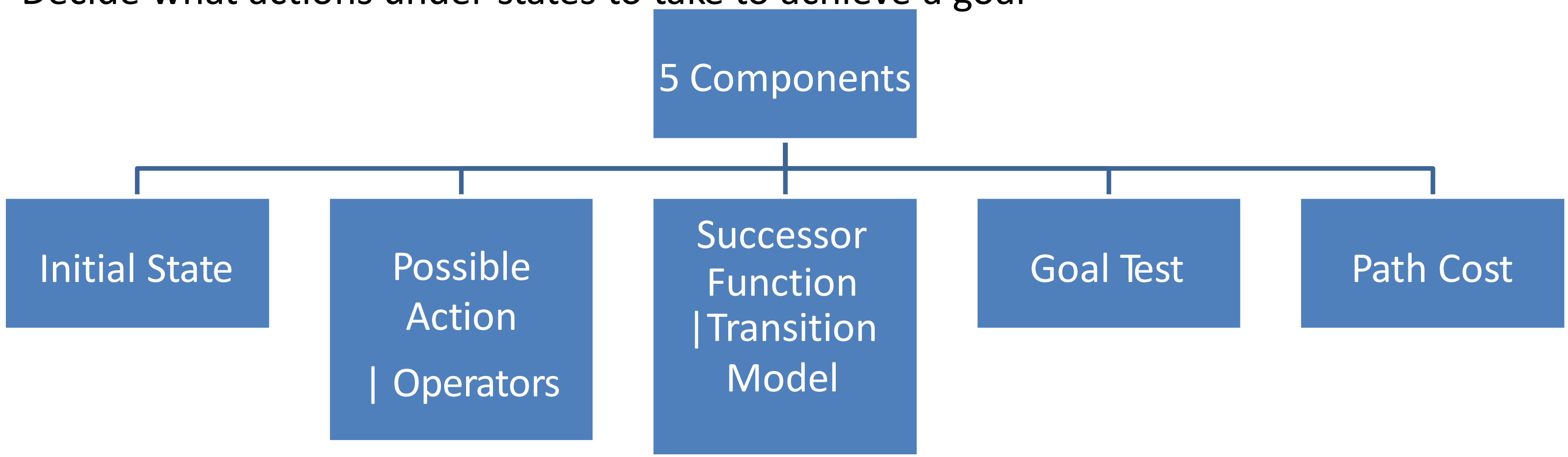
Execution Phase



Examine all sequence
Choose best |
Optimal

Problem Solving Agents – Problem Formulation

Abstraction Representation
Decide what actions under states to take to achieve a goal

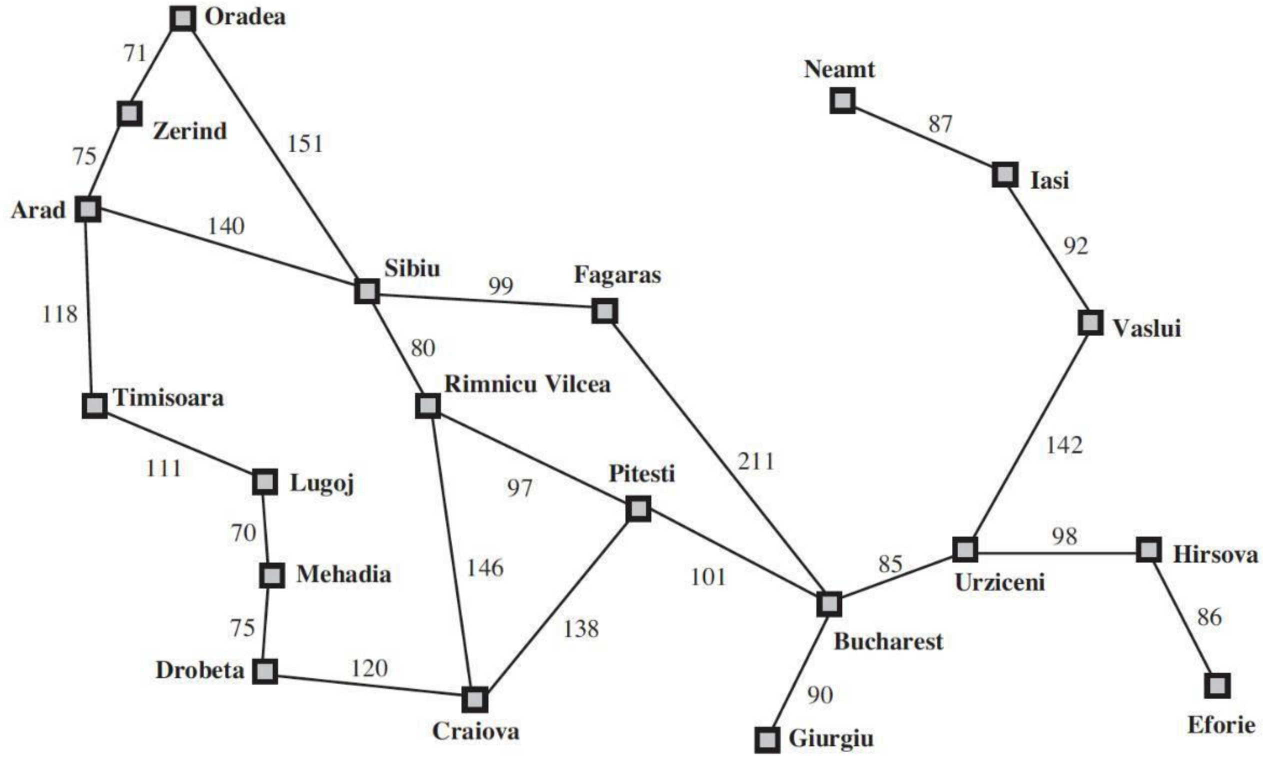


A function that assigns a numeric cost to each path. A path is a series of actions. Each action is given a cost depending on the problem.

Solution = Path Cost Function + Optimal Solution

Problem Solving Agents – Problem Formulation:

Book Example



Initial State –E.g., *In(Arad)*

Possible Actions – *ACTIONS(s) □ {Go(Sibiu), Go(Timisoara), Go(Zerind)}*

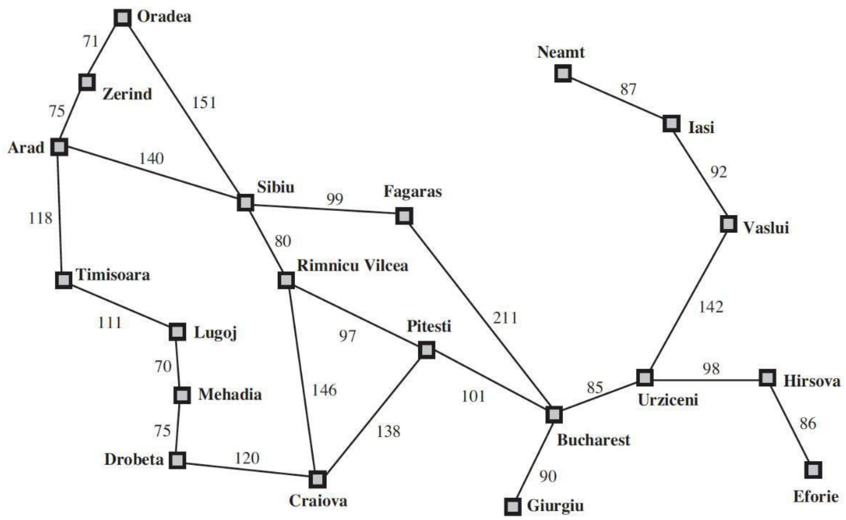
Transition Model – *RESULT(In(Arad), Go(Sibiu)) = In(Sibiu)*

Goal Test – *IsGoal(In(Bucharest)) = Yes*

Path Cost – *cost(In(Arad), go(Sibiu)) = 140 kms*

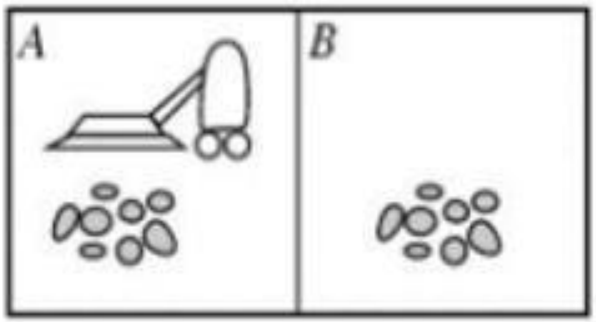
Example Problem Formulation

	Travelling Problem
Initial State	Based on the problem
Possible Actions	Take a flight | Train | Shop
Transition Model/ Successor Function	[A, Go(A->S)] = [S]
Goal Test	Is current = B (destination)
Path Cost	Cost + Time + Quality



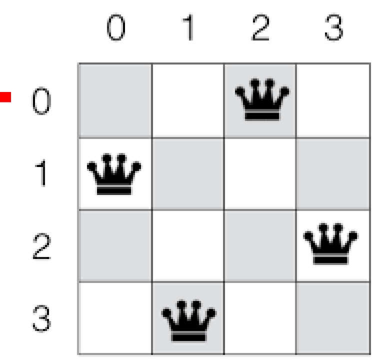
Example Problem Formulation

	Vacuum World
Initial State	Any
Possible Actions	[Move Left, Move Right, Suck, NoOps]
Transition Model/ Successor Function	[A, ML] = [B , Dirty] [A, ML] = [B, Clean]
Goal Test	Is all room clean? [A, Clean] [B, Clean]
Path Cost	No of steps in path



Example Problem Formulation

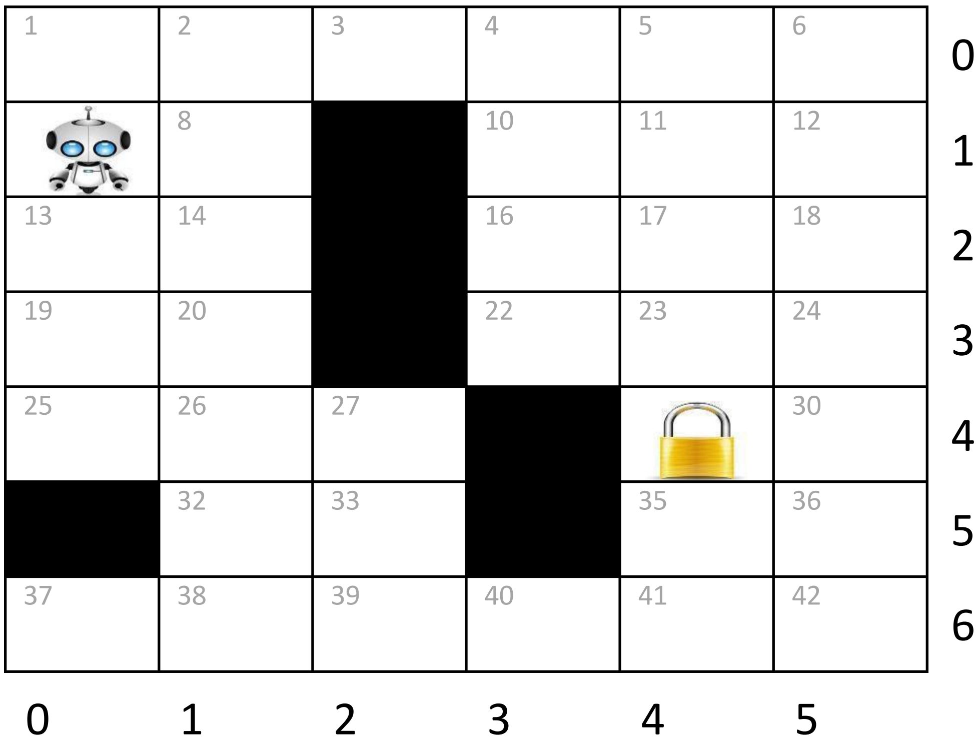
	N-Queen
Initial State	Empty | Partial | Full
Possible Actions	
Transition Model/ Successor Function	
Goal Test	
Path Cost	



board[r][c]

Path finding Robot

Successor Function Design



N-W-E-S

Graph Searching

- ➢Graph as state space (node = state, edge = action)
- ➢For example, game trees, mazes, ...

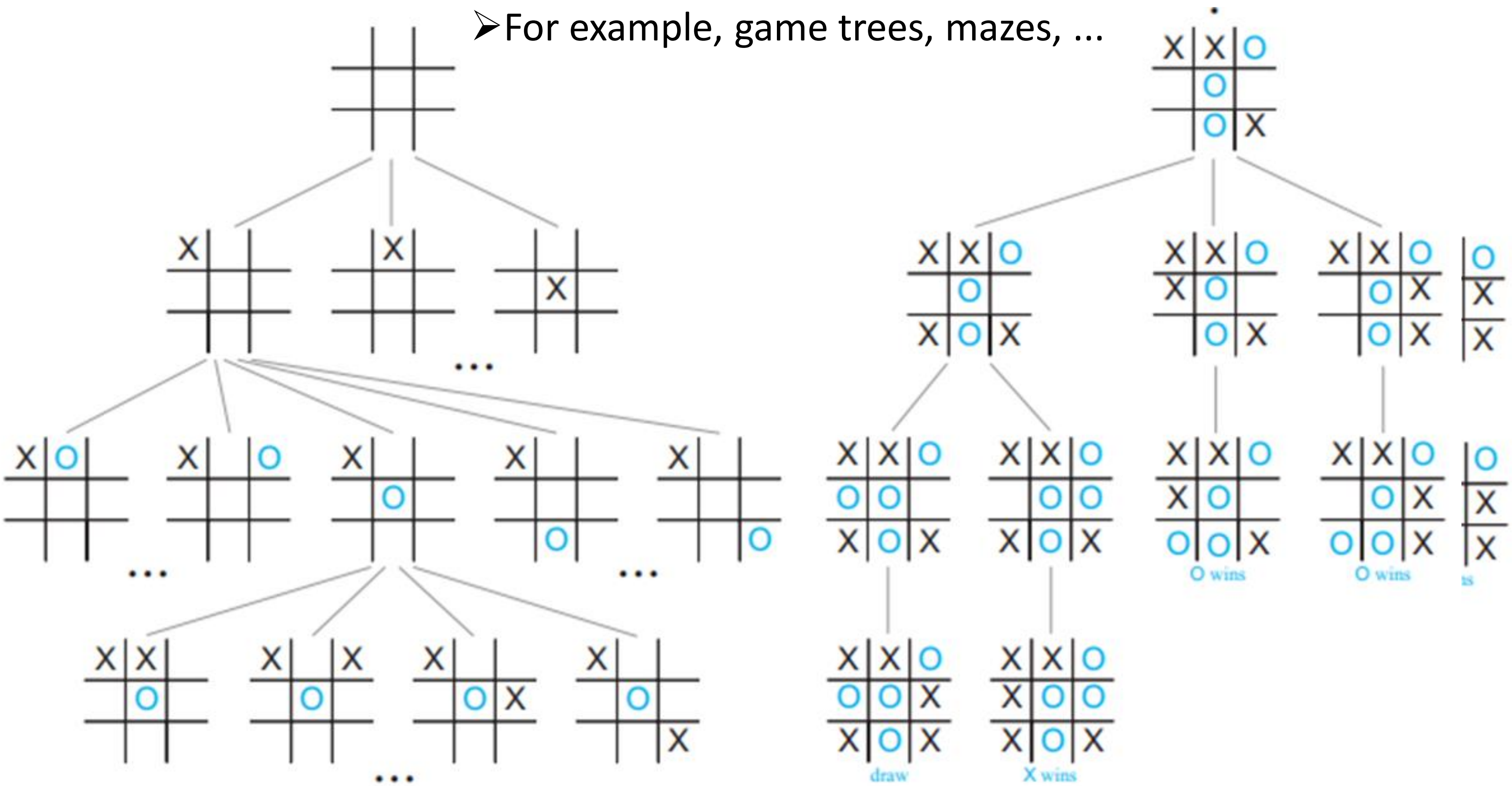


FIGURE 8 **Some of the Game Tree for Tic-Tac-Toe.**

Searching for Solutions

Choosing the current state, testing possible successor function, expanding current state to generate new state is called Traversal. Choice of which state to expand – Search Strategy

Search Strategy (under certainty)

Uninformed (will be discussed in next class)

BFS, DFS, UCS

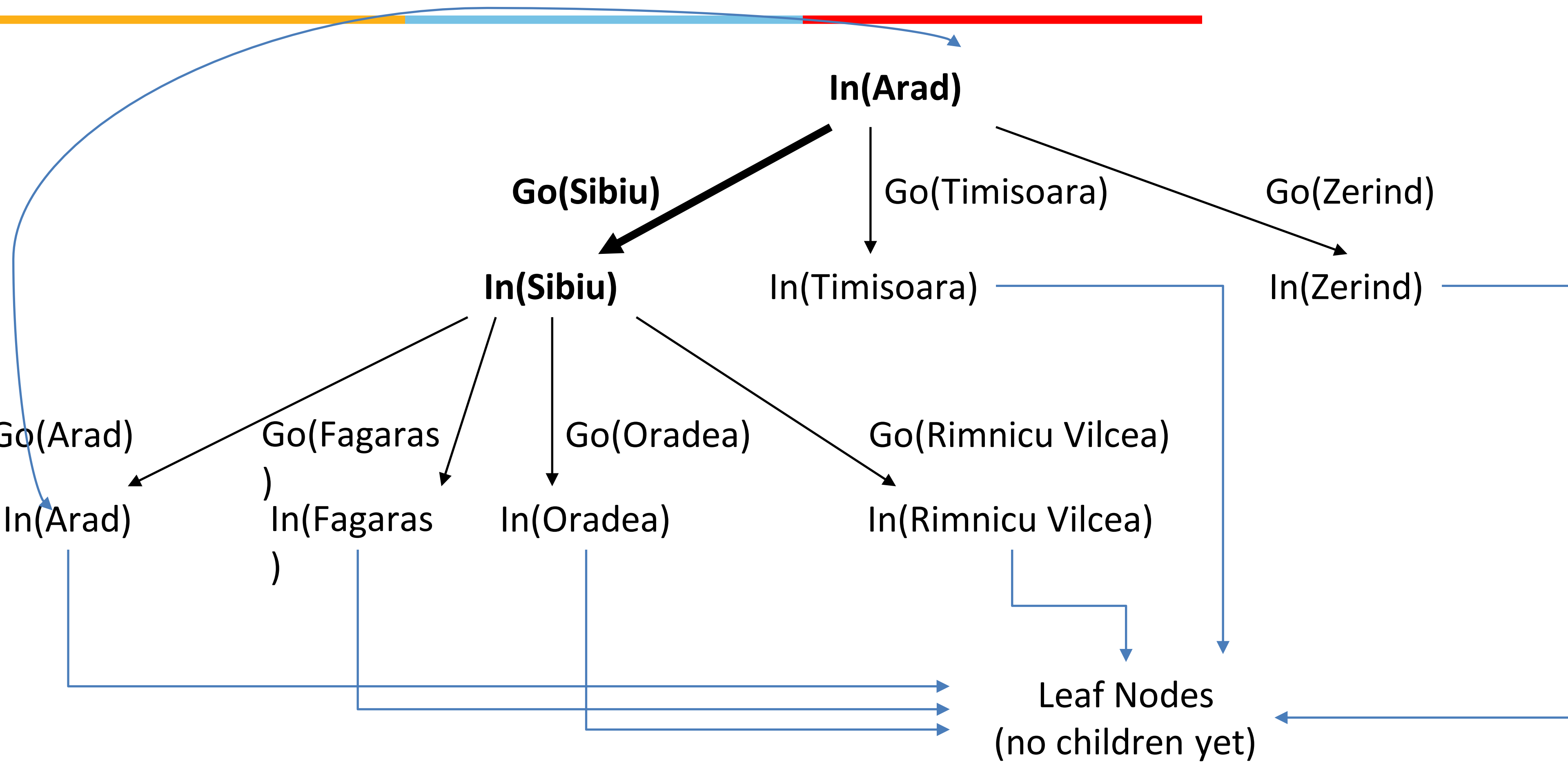
IDS, DLS

Bi-Directional

Informed (will be discussed in next class)

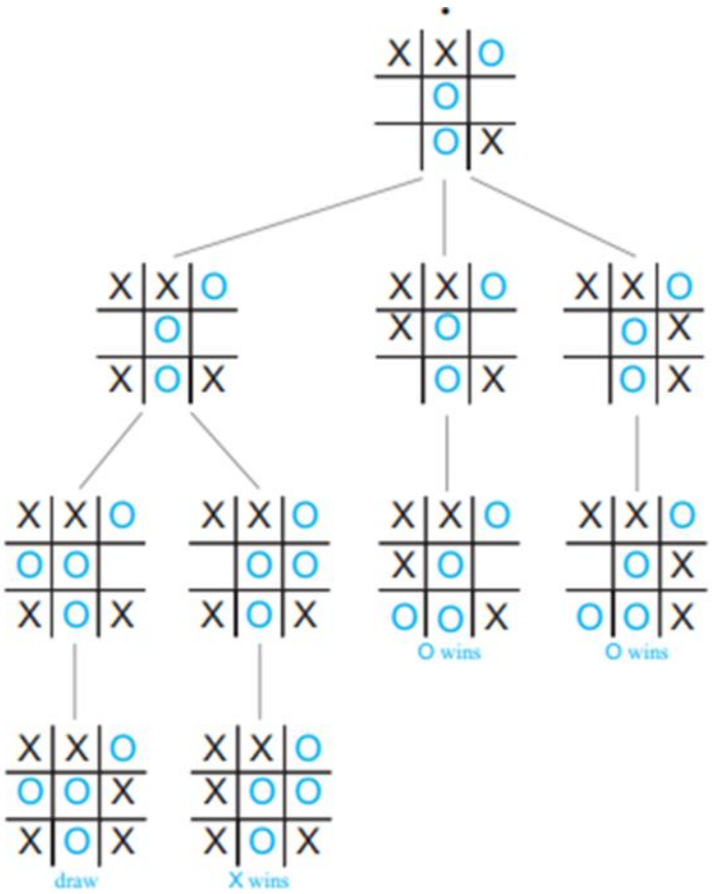
Best First Search

A*


In(Arad)
Go(Sibiu)
Go(Timisoara)
Go(Zerind)
In(Sibiu)
In(Timisoara)
In(Zerind)
Go(Arad)
Go(Fagaras)
Go(Oradea)
Go(Rimnicu Vilcea)
In(Arad)
In(Fagaras)
In(Oradea)
In(Rimnicu Vilcea)
Leaf Nodes
(no children yet)

Graph Searching

- BFS and DFS each search the state space for a best move.
- If the search is exhaustive they will find the same solution, but if there is a time limit and the search space is large...
- DFS explores a few possible moves, looking at the effects far in the future
- BFS explores many solutions but only sees effects in the near future (often finds shorter solutions)



Next Class Plan

- Uninformed Search Algorithms
  - BFS vs DFS – An overview
  - Uniform Cost Search
  - Iterative Depth First Search
  - Notion of Bi-Directional Search
- Informed Search Algorithms
  - Greedy Best First search
  - A* Search (Start)

Required Reading: AIMA - Chapter #1, 2, 3.1, 3.2, 3.3

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials