



Artificial and Computational Intelligence

AIMLCZG557

Contributors & Designers of document content : Cluster Course Faculty Team

M2 : : Problem Solving Agent using Search



BITS Pilani

Pilani Campus

Artificial and Computational Intelligence

Disclaimer and Acknowledgement



- Few content for these slides may have been obtained from prescribed books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online.
- .I have provided source information wherever necessary
- This is not a full fledged reading materials. Students are requested to refer to the textbook w.r.t detailed content of the presentation deck that is expected to be shared over e-learning portal - taxilla.
- I have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation
- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP: Prof.Raja vadhana, Prof. Indumathi, Prof.Sangeetha
- From BITS Oncampus & External : Mr.Santosh GSK

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

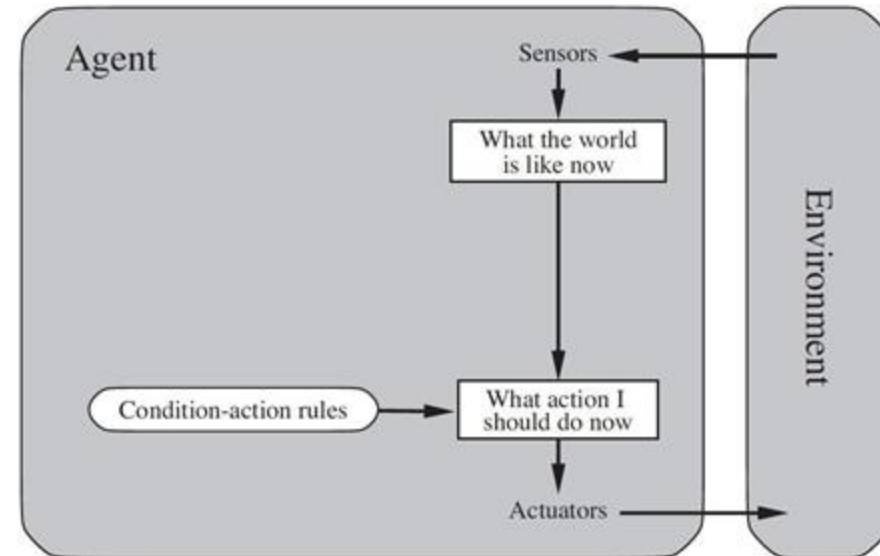
Agent Architectures



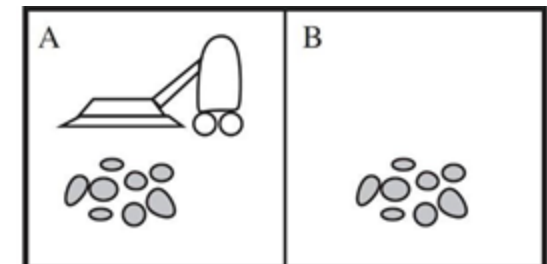
Simple Reflex Agent

```
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
```

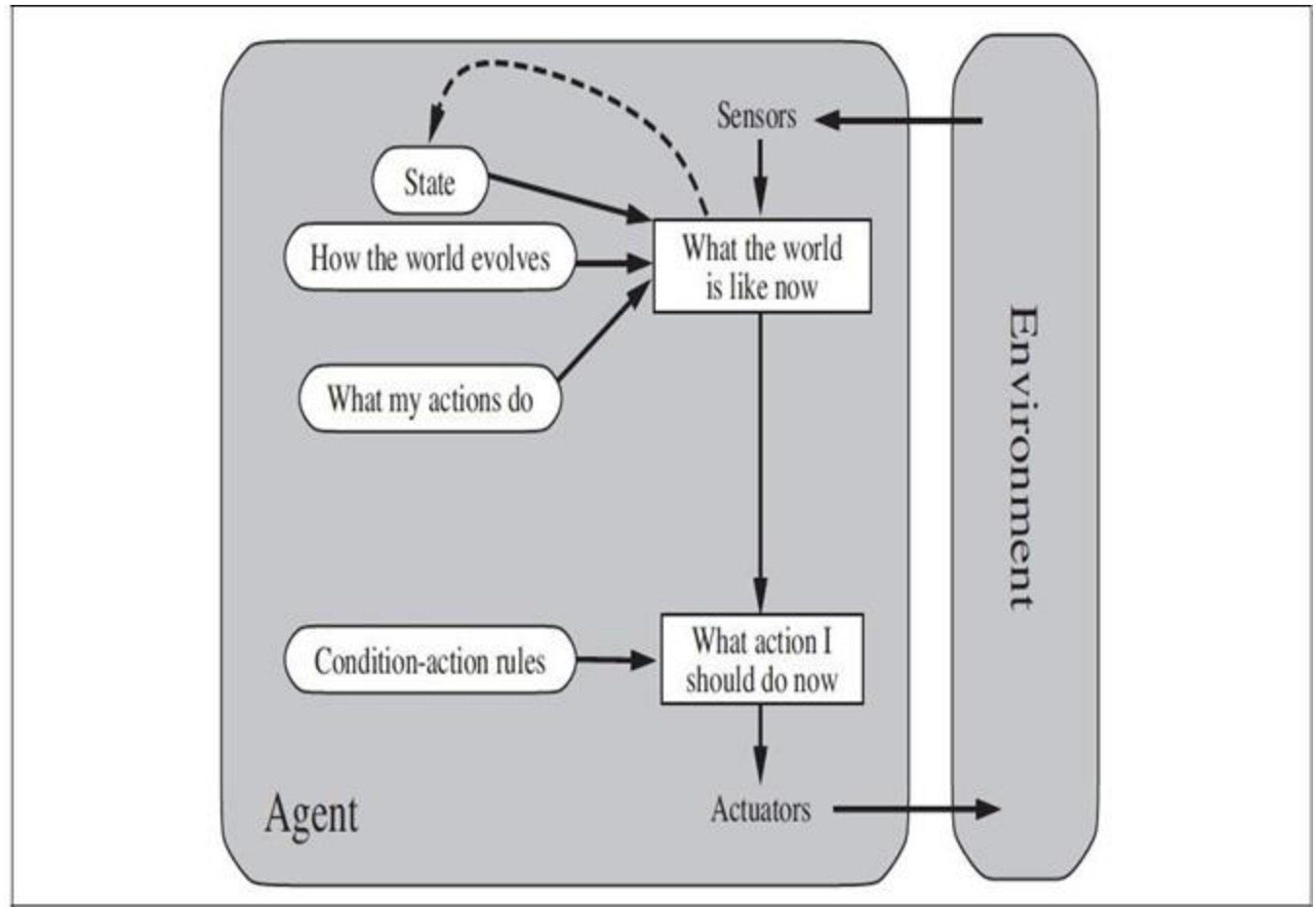
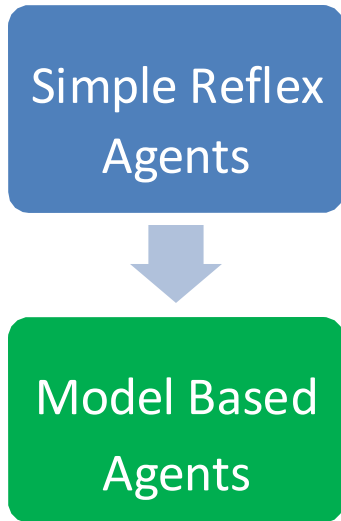
```
function REFLEX-VACUUM-AGENT( [location, status]) returns an action
  if status = Dirty then return Suck
  else if location = A then return Right
  else if location = B then return Left
```



Simple Reflex
Agents



Model based Agent



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*)

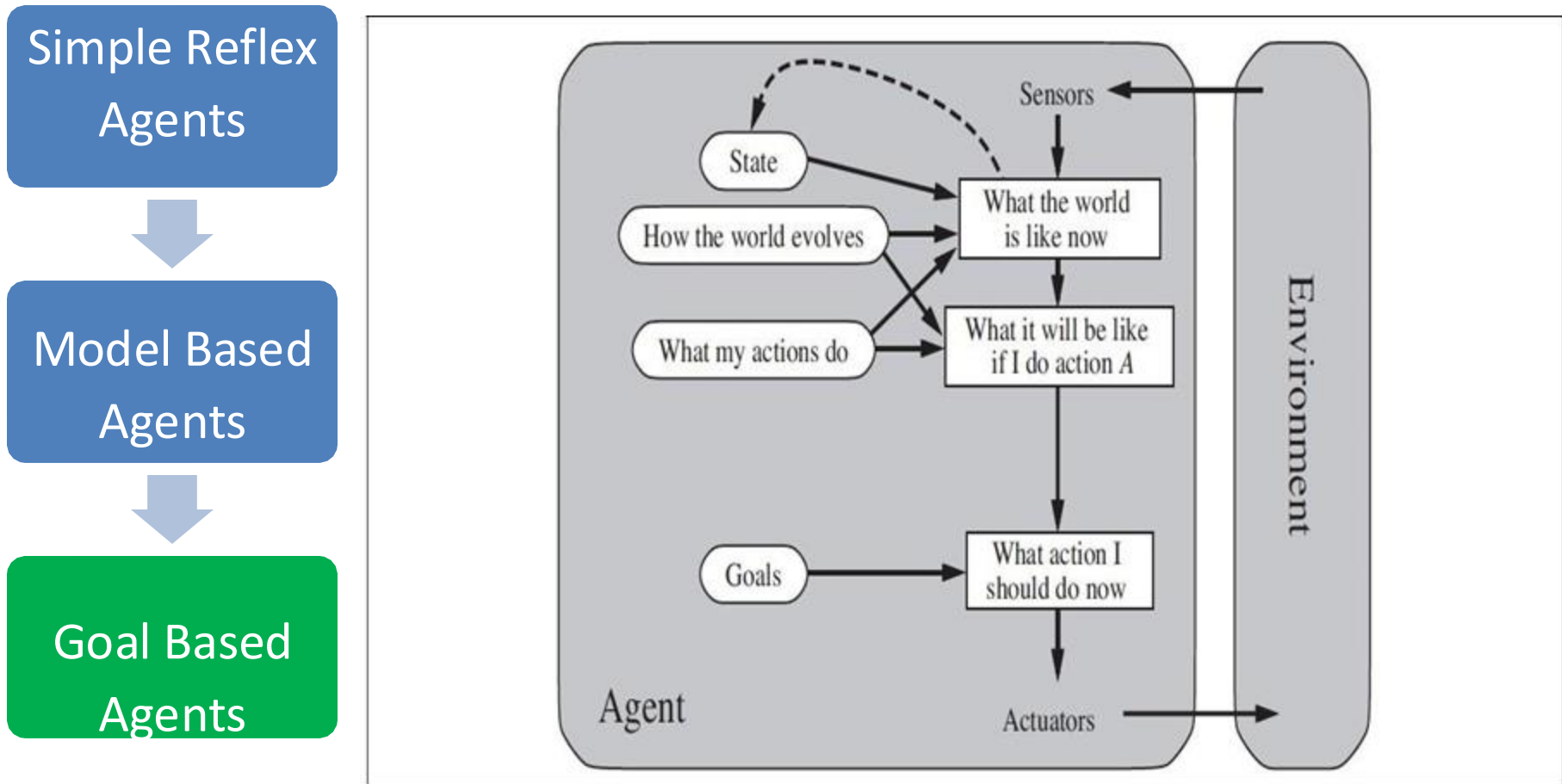
rule ← RULE-MATCH(*state*, *rules*)

action ← *rule*.ACTION

return *action*

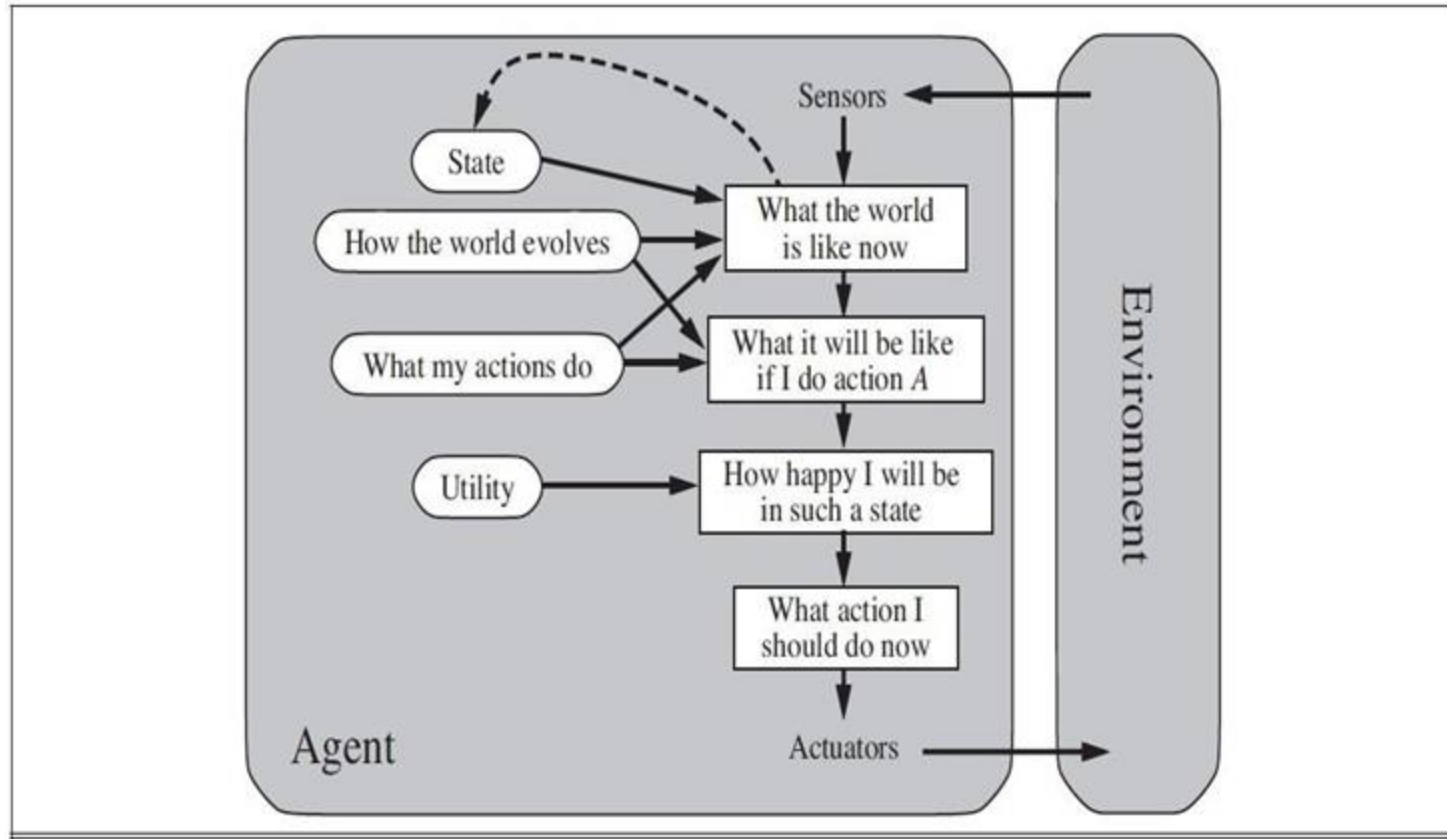
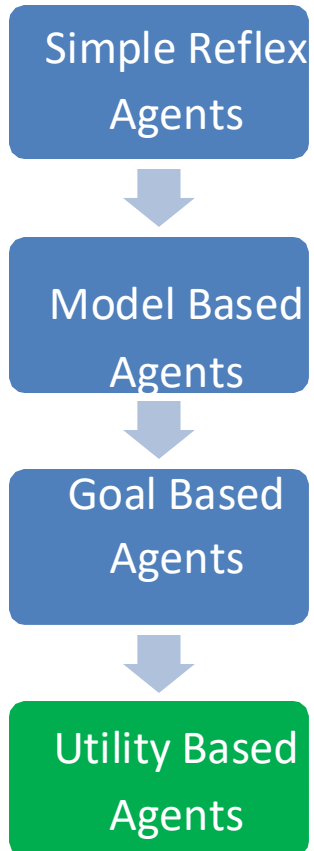
Agent Architectures

Goal based Agent



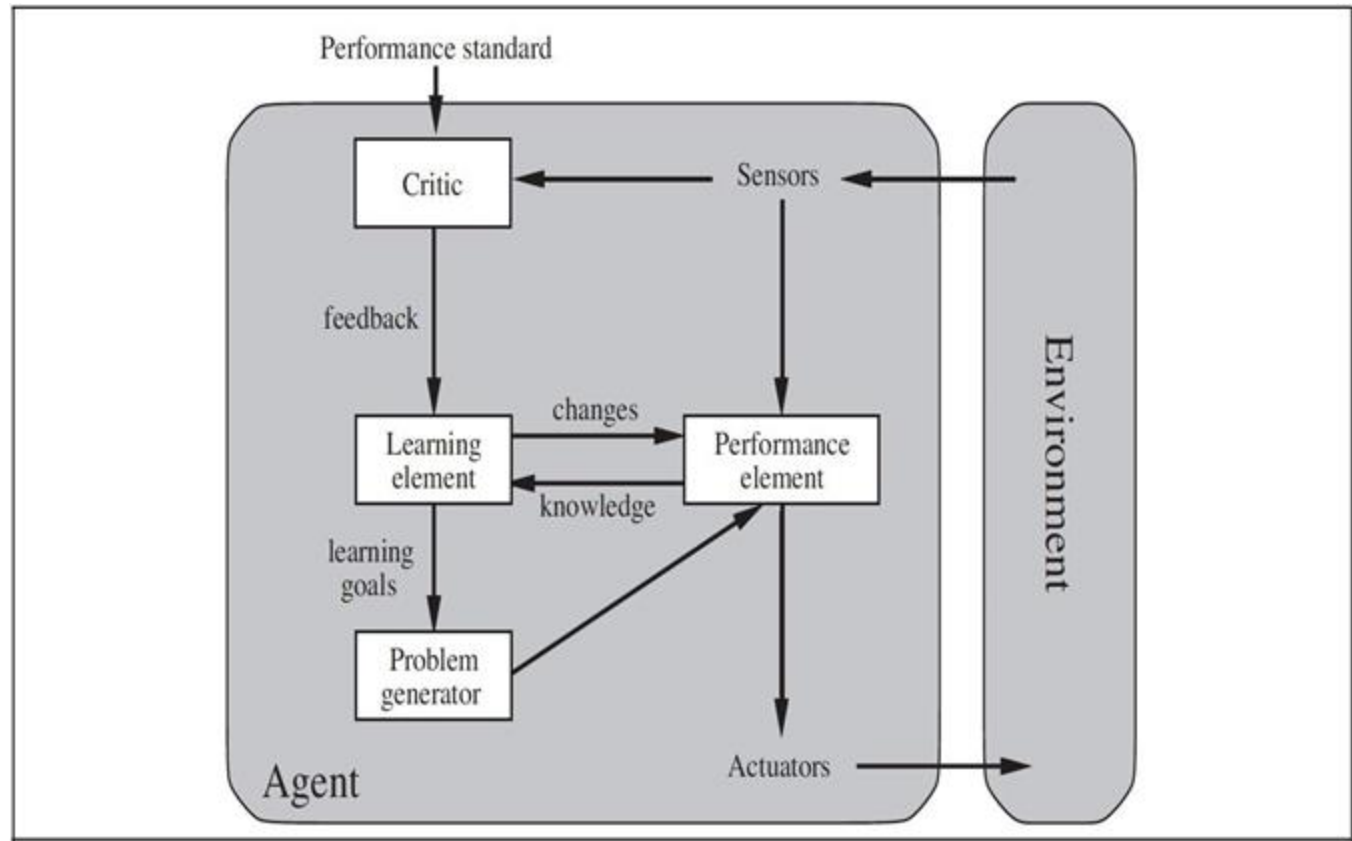
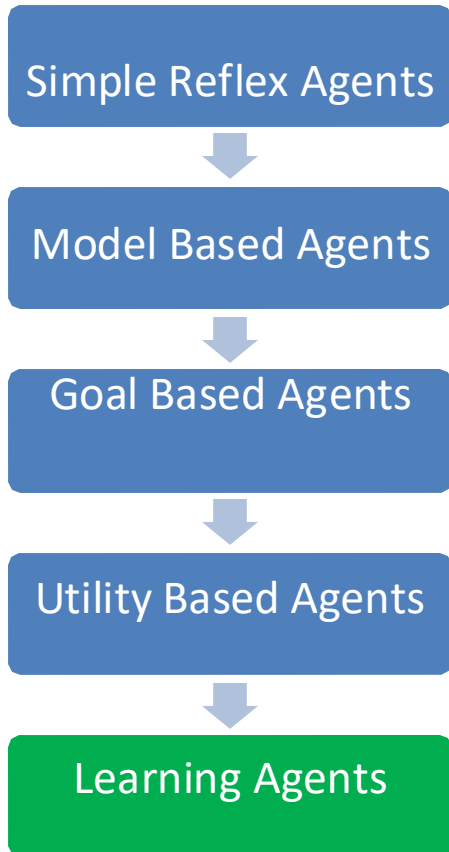
Agent Architectures

Utility based Agent



Agent Architectures

Learning Agent



Role of Learning



Performance Element – taking a decision of action based on percepts

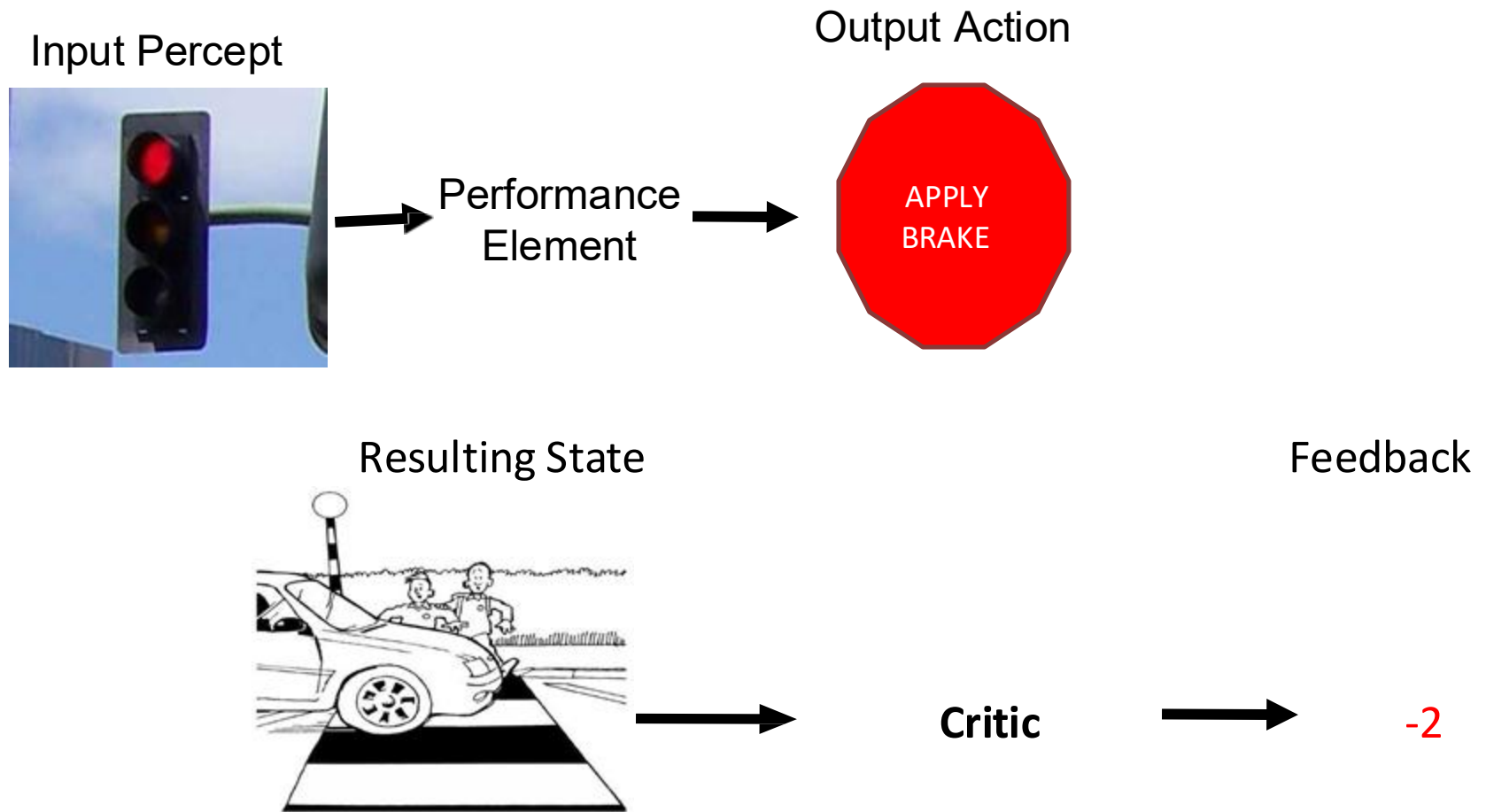
Learning Element – Make the performance element select better actions such that the utility function is optimized

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

Role of Learning

Agents that improve their performance by learning from their own experiences



Role of Learning

Input Percept



Possible Actions

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

Random



Selected Action

Change Gear to Lower



Role of Learning

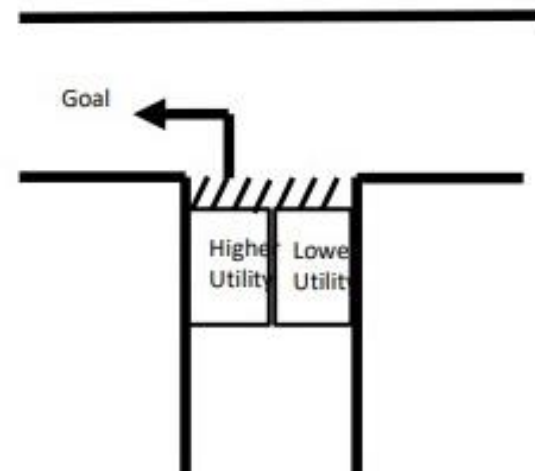
Performance Element – Takes decision on action based on percept

$$f(\text{red signal, distance}) = 15k \text{ N brake}$$

$$\text{distance} = f'(\text{percept sequence})$$

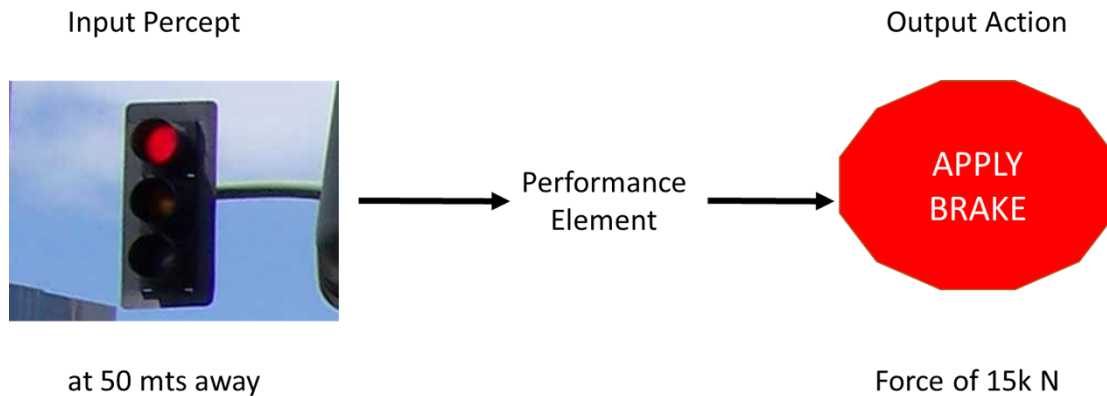
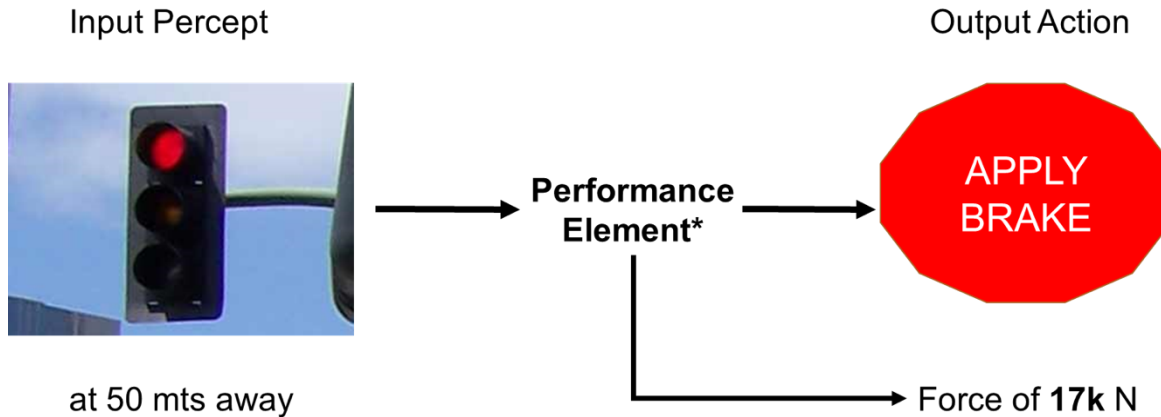
$$f(\text{percepts, distance, raining})$$

- $f(\text{state}_0, \text{actionA}) = 0.83,$
- $f(\text{state}_0, \text{actionB}) = 0.45$



Role of Learning

Learning : Supervised Vs Unsupervised Vs Reinforcement



+ Critic Feedback -2

Role of Learning

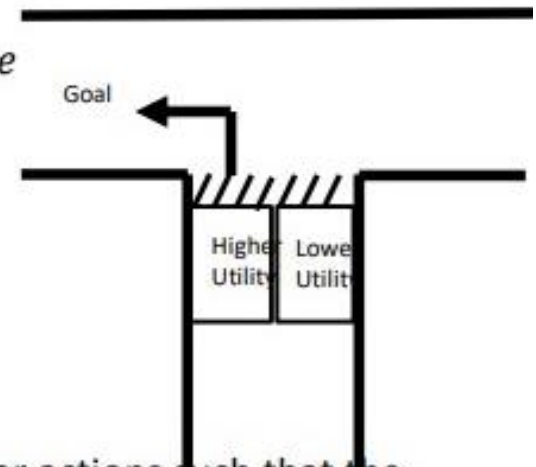
Performance Element – Takes decision on action based on percept

$$f(\text{red signal, distance}) = 15k \text{ N brake}$$

$$\text{distance} = f'(\text{percept sequence})$$

$$f(\text{percepts, distance, raining})$$

- $f(\text{state}_0, \text{actionA}) = 0.83,$
- $f(\text{state}_0, \text{actionB}) = 0.45$



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

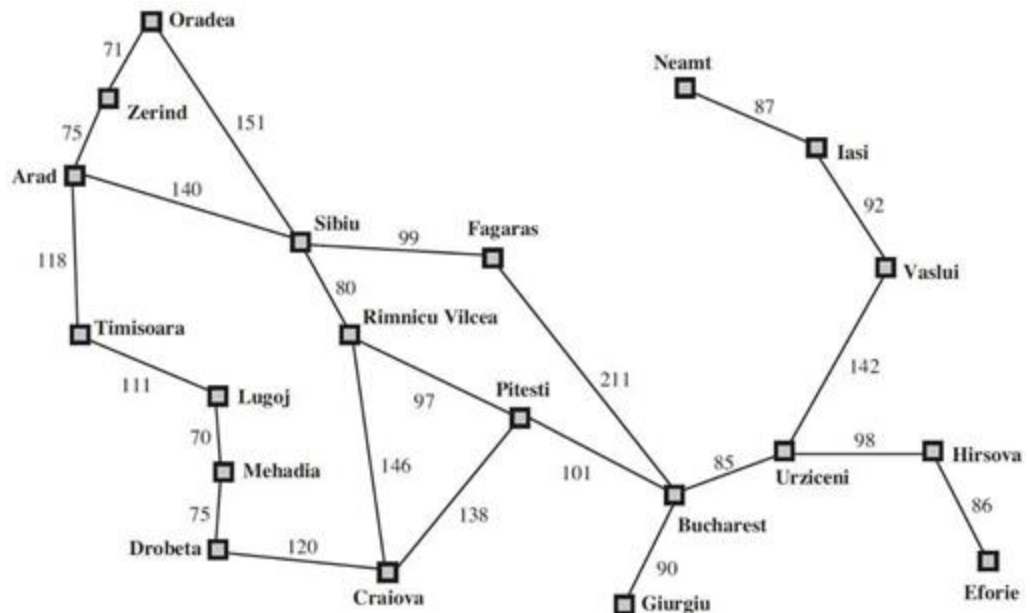
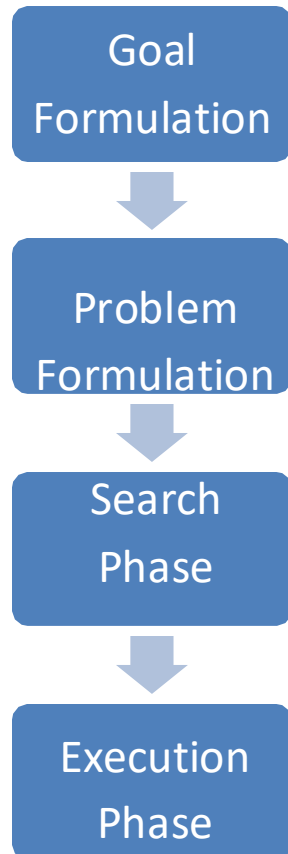
Problem Formulation

Problem Solving Agents

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

Phases of Solution Search by PSA

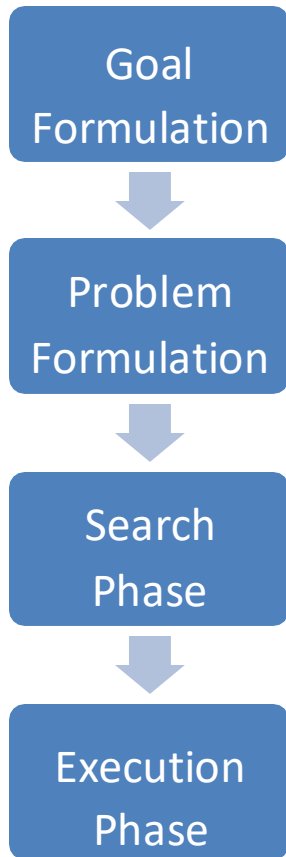
Optimizes the Objective (Local | Global) Limits the Actions



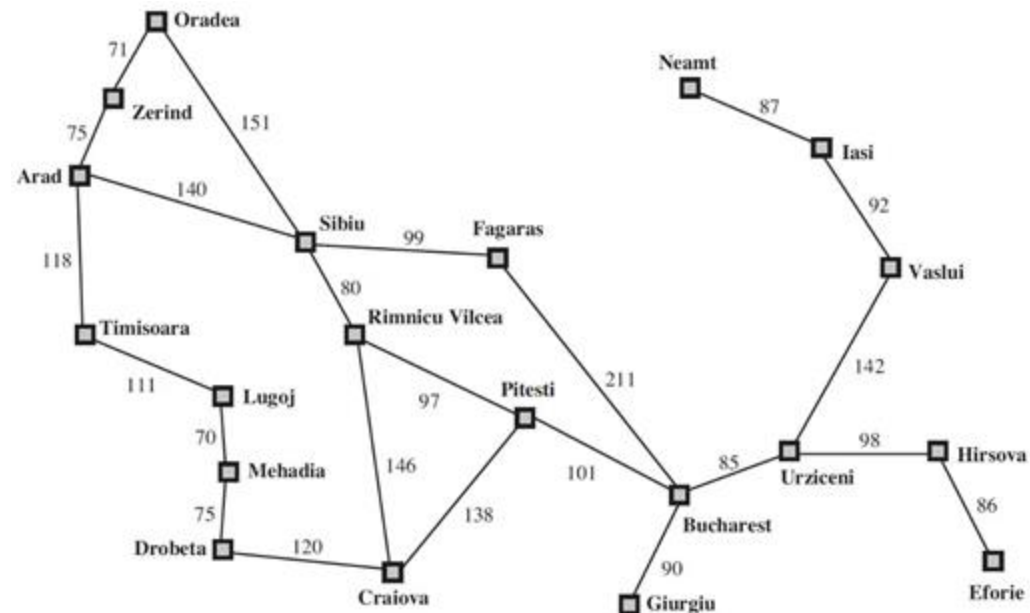
Problem Solving Agents



Phases of Solution Search by PSA



State Space Creations [in the path of Goal]
Lists the Actions

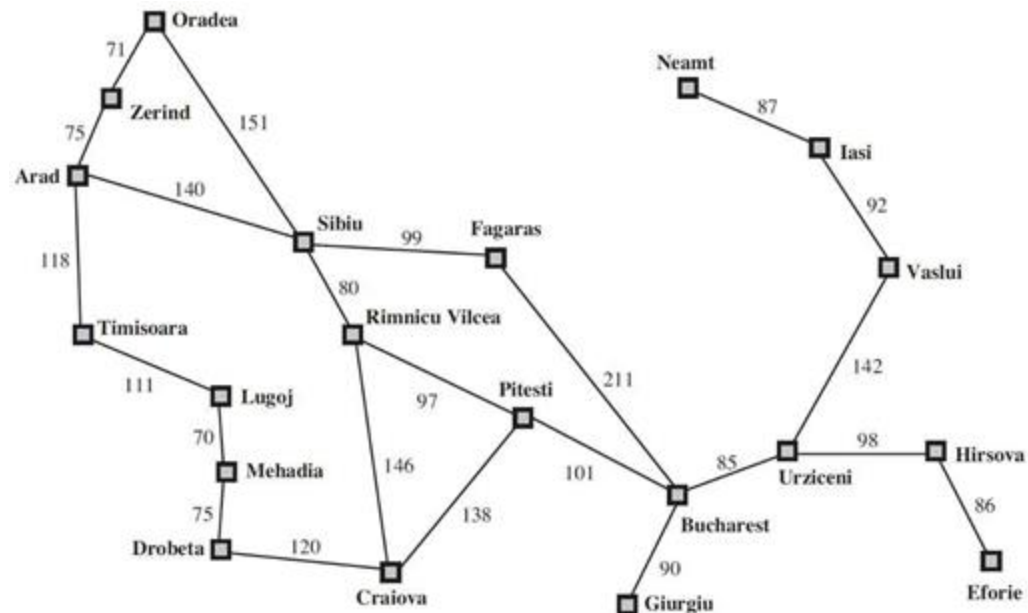
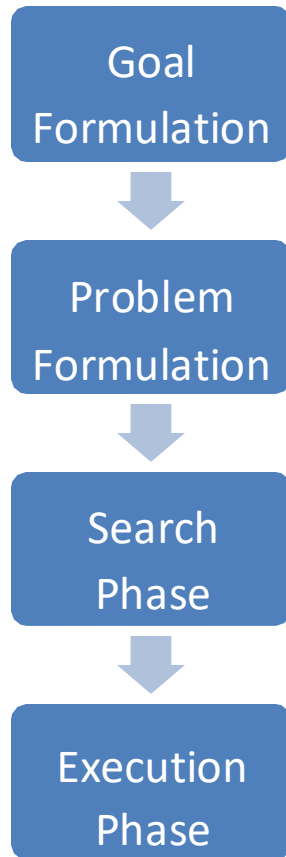


Problem Solving Agents

Phases of Solution Search by PSA

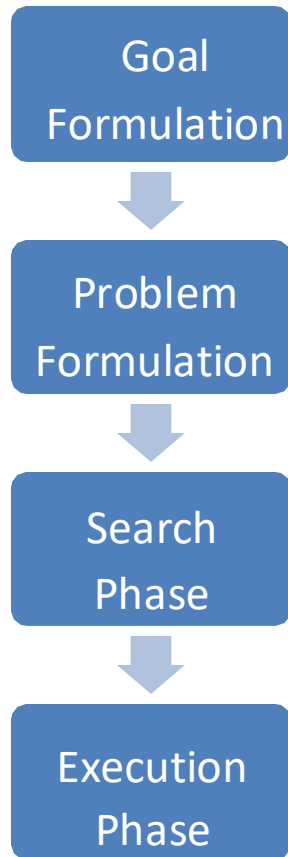
Assumptions – Environment :

- Static
- Observable Discrete
- Deterministic

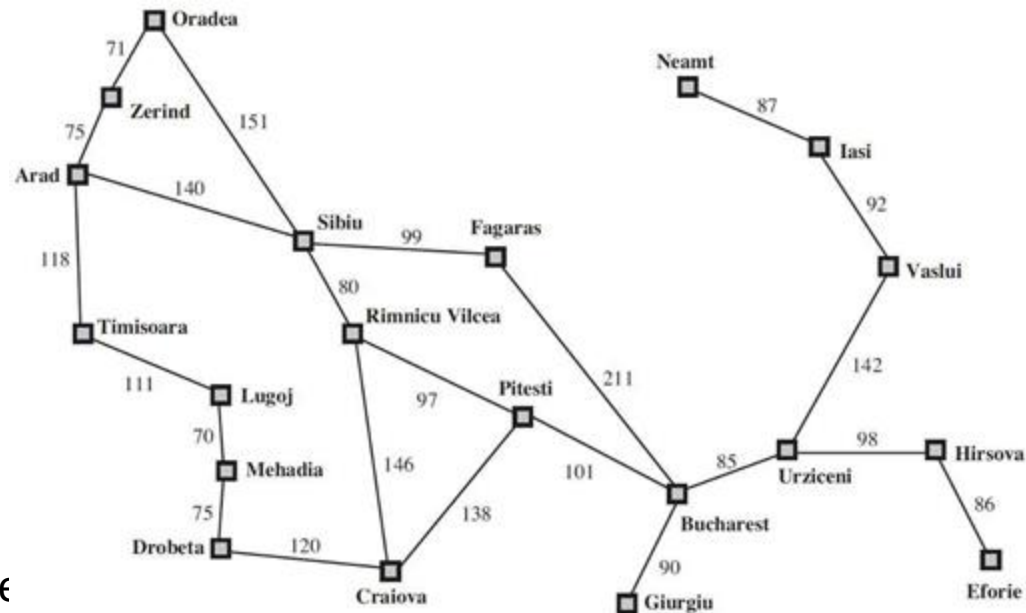


Problem Solving Agents

Phases of Solution Search



Examine all sequence
Choose best | Optimal

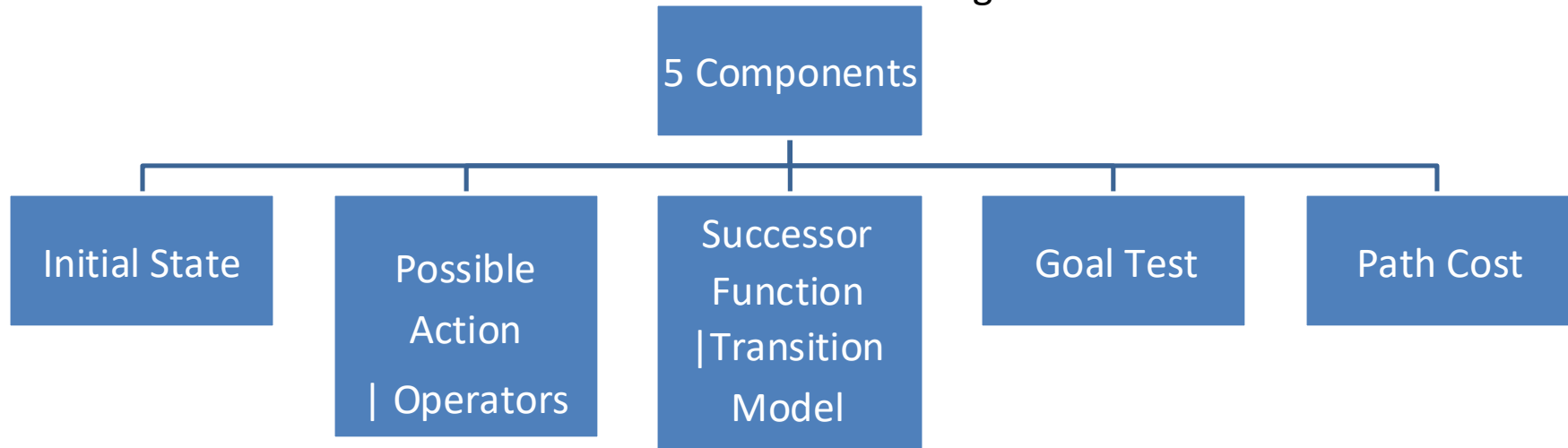




Problem Solving Agents – Problem Formulation

Abstraction Representation

Decide what actions under states to take to achieve a goal

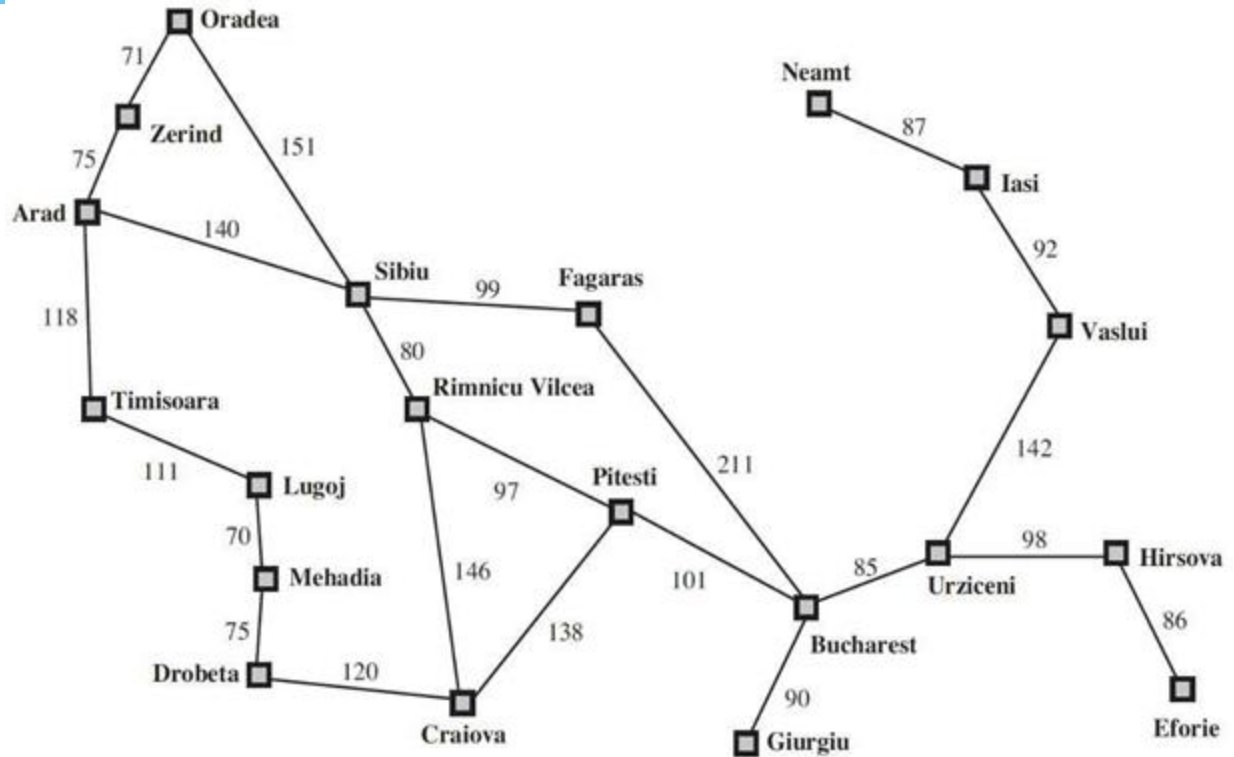


A function that assigns a numeric cost to each path. A path is a series of actions.

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 – $ACTIONS(s) \rightarrow \{Go(Sibiu), Go(Timisoara), Go(Zerind)\}$

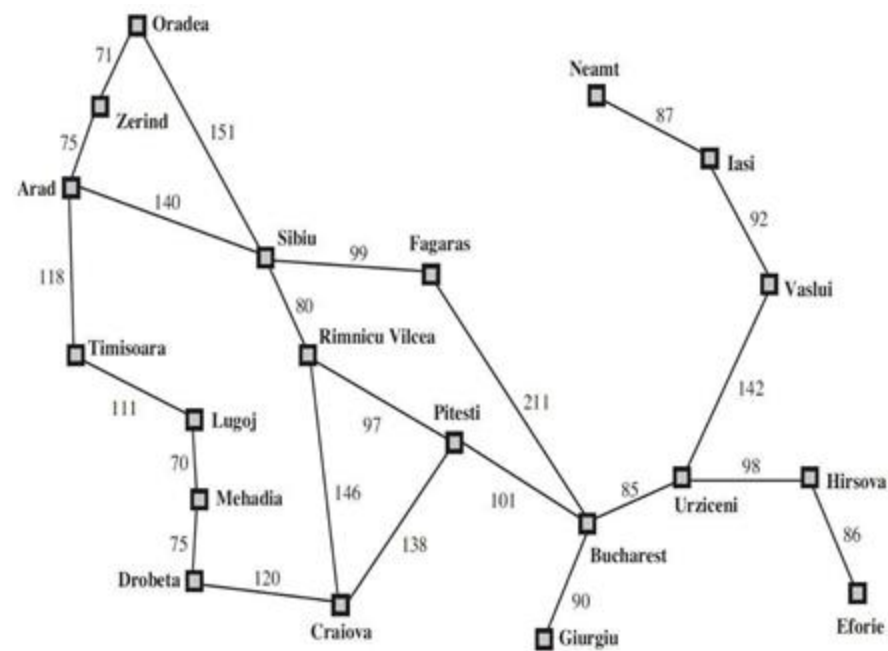
Transition Model – $RESULT(In(Arad), Go(Sibiu)) = In(Sibiu)$

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

Path Cost – $cost(In(Arad), go(Sibiu)) = 140 \text{ kms}$

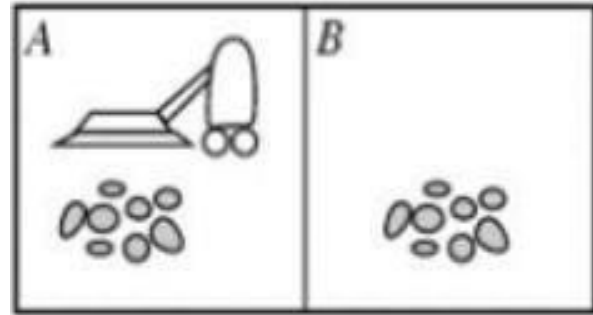
Example Problem Formulation

	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	

	0	1	2	3
0			♔	
1	♔			
2				♔
3		♔		

board[r][c]

Path finding Robot

Successor Function Design

1	2	3	4	5	6	0
	8		10	11	12	1
13	14		16	17	18	2
19	20		22	23	24	3
25	26	27			30	4
	32	33		35	36	5
37	38	39	40	41	42	6
0	1	2	3	4	5	

N-W-E-S

Graph Searching

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

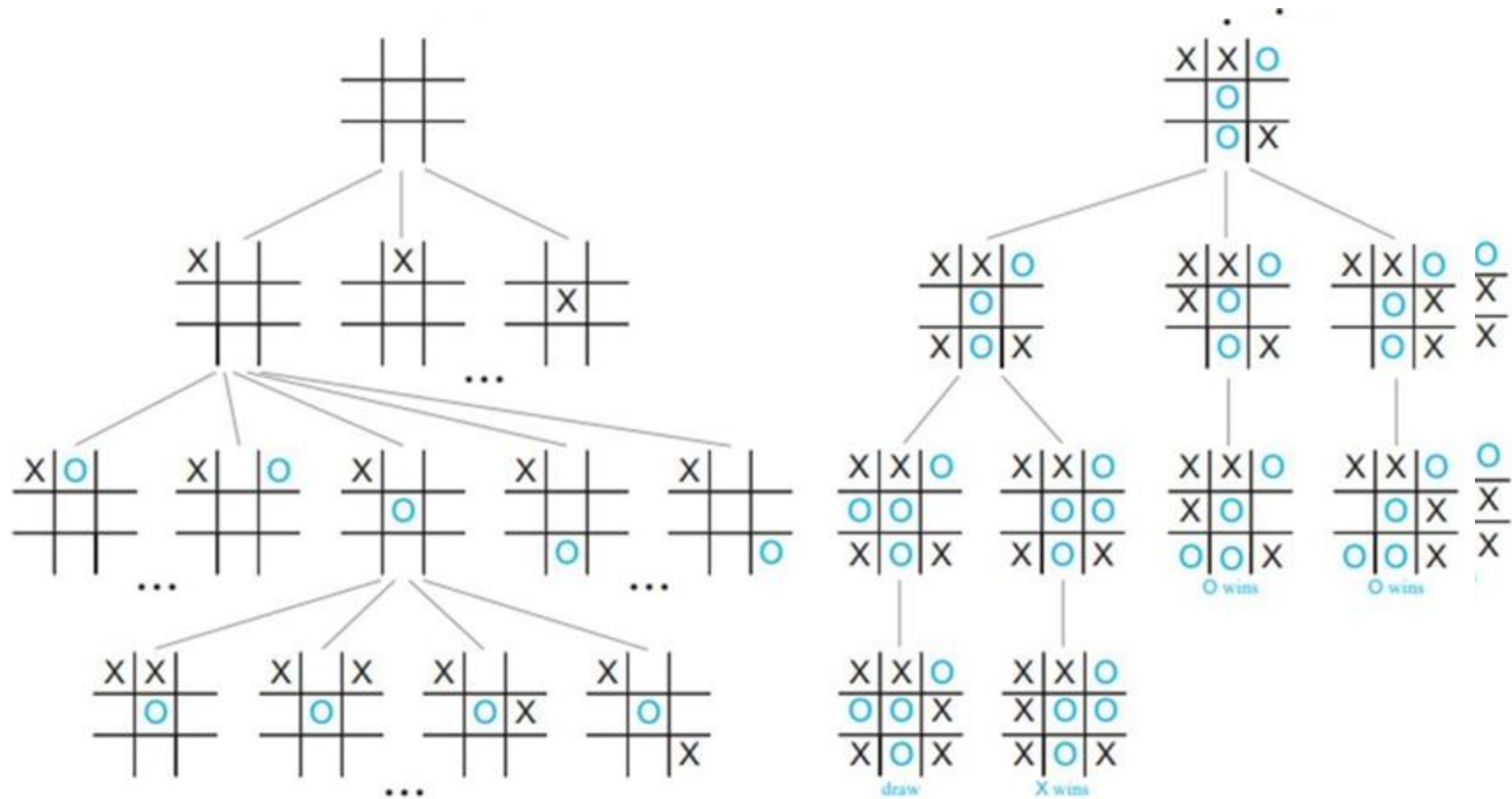
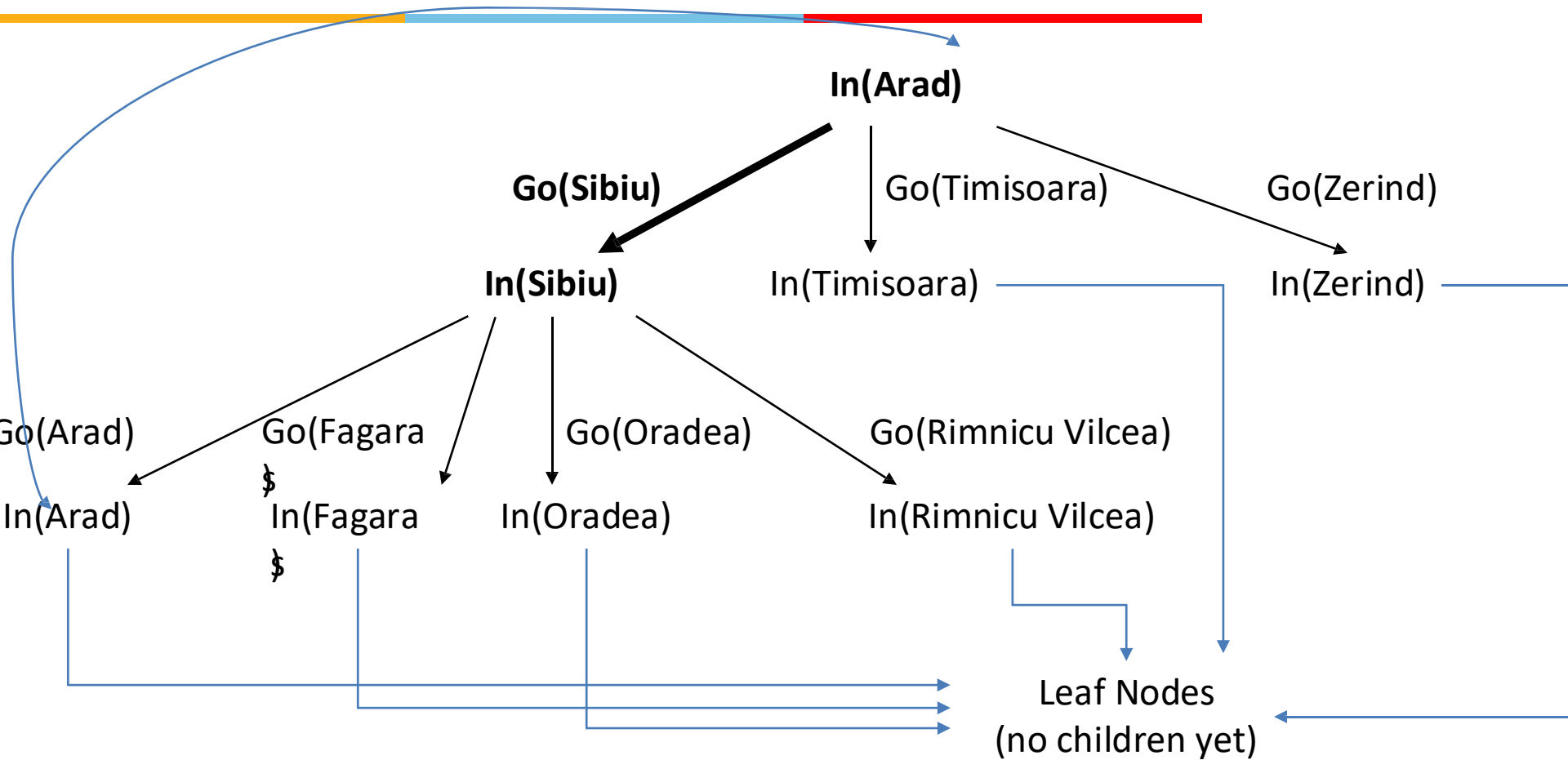


FIGURE 8 Some of the Game Tree for Tic-Tac-Toe.

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





Next Class Plan

- Uninformed Search Algorithms
 - BFS vs DFS – An overview
 - Uniform Cost Search
 - Iterative Depth First 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