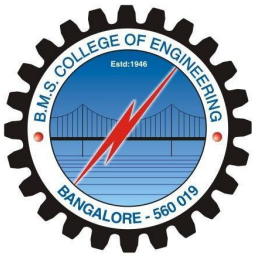
Artificial Intelligence
22CS5PCAIN

Course Instructor

Dr. Umadevi V

Department of CSE, BMSCE



Unit-1

Unit-1: Definition, Agents: Agents and environment, Concept of Rationality, The nature of environment, The structure of agents. Problem-solving: Problem-solving agents, Example problems, Searching for Solutions.

What is AI ?

Four Approaches to AI

Thinking Humanly: Systems that think like Humans	Thinking Rationally: Systems that think Rationally
Acting Humanly: Systems that act like Humans	Acting Rationally: Systems that act Rationally

The “approach of AI” refers to the general philosophy or strategy that is used
to build and design artificial intelligence (AI) systems.

Thinking Humanly:

- This approach focuses on building artificial intelligence systems that can think like a human. The goal is to create systems that can **understand** human language, emotions, and culture and can interact with humans in a natural way.
- This approach is mainly used in the development of conversational AI systems, such as chatbots and virtual assistants, that need to understand and respond to natural language input from humans.
- Building systems that function *internally* in some way similar to human mind

Example: Siri, Alexa, Chatbots

Thinking Rationally:

- This approach focuses on building artificial intelligence systems that can reason logically and **make decisions based on information** and rules. The goal is to create systems that can solve problems and make decisions in a way that is consistent with the principles of rational thinking.
- This approach is used in a wide range of applications, including decision-making, planning, and problem-solving. For example, expert systems, recommendation systems, and optimization algorithms.
- System doing the "Right Thing" given what it knows

Example: Recommendation systems, Expert Systems, Optimization Algorithms

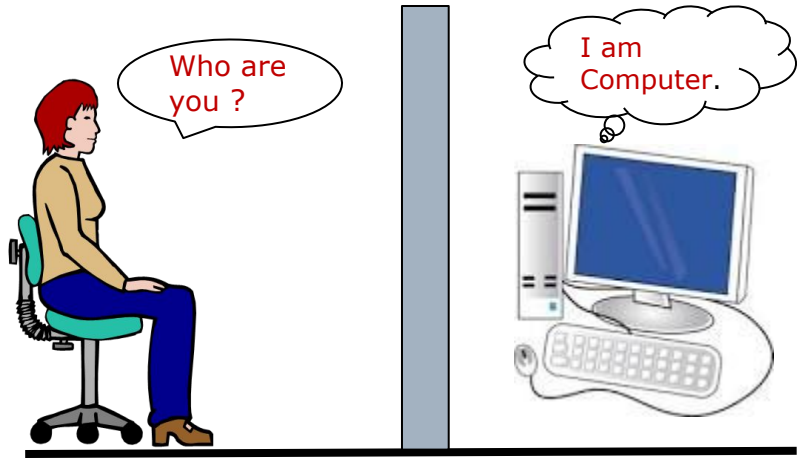
Acting Humanly

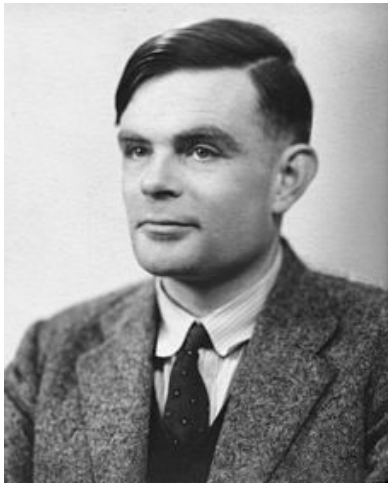
- This approach focuses on building artificial intelligence systems that can act like humans. The goal is to create systems that can perform tasks such as recognizing speech, recognizing images, and controlling robots in a human-like manner.
- This approach is mainly used in computer vision and robotics, where the goal is to create systems that can perceive and **interact** with the physical world in a human-like manner.
- How can knowledge be represented logically, and how can a system draw deductions?

Example: Self-driving cars, Facial Recognition systems

Acting Humanly - Turing Test Approach

- The Turing test is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.
- The Computer in interrogated by a human via a teletype.
- It passes if the human cannot tell if there is a computer or human at the other end.





Alan Turing

The Turing Test, proposed by Alan Turing (1950), was designed to provide a satisfactory operational definition of intelligence

Acting Rationally:

- This approach focuses on building artificial intelligence systems that can act rationally. The goal is to create systems that can make decisions and take actions that are consistent with the principles of rational thinking and that achieve their goals efficiently and effectively.

- This approach is mainly used in artificial intelligence systems that need to make decisions and take actions to achieve their goals in a rational and efficient manner. For example, autonomous agents and reinforcement learning algorithms.

Example: Reinforcement learning algorithms, Games AI

Summarizing Four Approaches to AI

Thinking Humanly: Systems that think like Humans **Example: Siri, Alexa, Chatbots** Thinking humanly — cognitive modeling. Systems should solve problems the same way humans do.	Thinking Rationally: Systems that think Rationally **Example: Recommendation systems, Expert Systems, Optimization Algorithms** Thinking rationally — **the use of logic**.
Acting Humanly: Systems that act like Humans **Example: Self-driving cars, Facial Recognition systems** Acting humanly — the Turing Test approach.	Acting Rationally: Systems that act Rationally **Example: Reinforcement learning algorithms, Games AI** Acting rationally — the study of rational agents: agents that **maximize the expected** value of their performance measure given what they currently know.

Table 1: Four Categories of AI Definitions

	Human Behaviour	Rational Behaviour
Thinking (Mental Process)	**1. Thinking Humanly** Machines that think intelligently like humans	**3. Thinking Rationally** Machines that think rationally
Acting (Action)	**2. Acting Humanly** Machines that perform activities that human consider intelligent	**4. Acting Rationally** Machines that act rationally

Question

Artificial Intelligence is about_____.

- Playing a game on Computer
- Making a machine Intelligent
- Programming on Machine with your Own Intelligence
- Putting your intelligence in Machine

Question

Artificial Intelligence is about_____.

- Playing a game on Computer
- **Making a machine Intelligent**
- Programming on Machine with your Own Intelligence
- Putting your intelligence in Machine

Question

To evaluate whether machine is acting humanly ……. is used?

- Turing test
- Cognitive modelling
- Laws of thoughts
- All of these

Question

To evaluate whether machine is acting humanly ……. is used?

- **Turing test**
- Cognitive modelling
- Laws of thoughts
- All of these

Question

To evaluate whether machine is thinking humanly ……. is used?

- Turing test
- Cognitive modelling
- Laws of thoughts
- All of these

Question

To evaluate whether machine is thinking humanly ..... is used?

- Turing test
- **Cognitive modelling**
- Laws of thoughts
- All of these

Question

To evaluate whether machine is thinking rationally ..... is used?

- Turing test
- Cognitive modelling
- Laws of thoughts
- All of these

Question

To evaluate whether machine is thinking rationally ..... is used?

- Turing test
- Cognitive modelling
- **Laws of thoughts**
- All of these

Agents

Agents: Agents and environment, Concept of Rationality, The nature of environment, The structure of agents.

Agents

- Perceive the environment through sensors (→ Percepts)
- Act upon the environment through actuators (→ Actions)

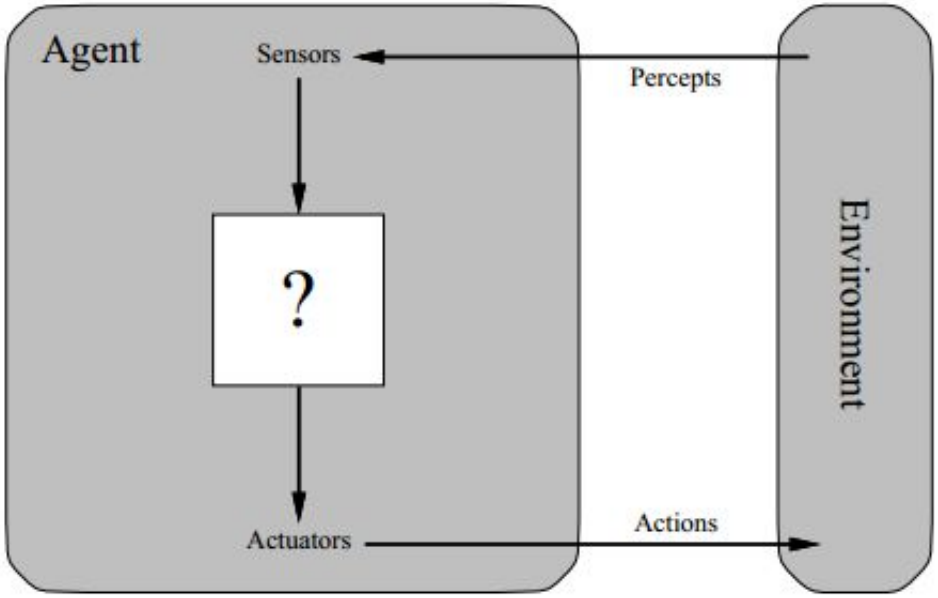


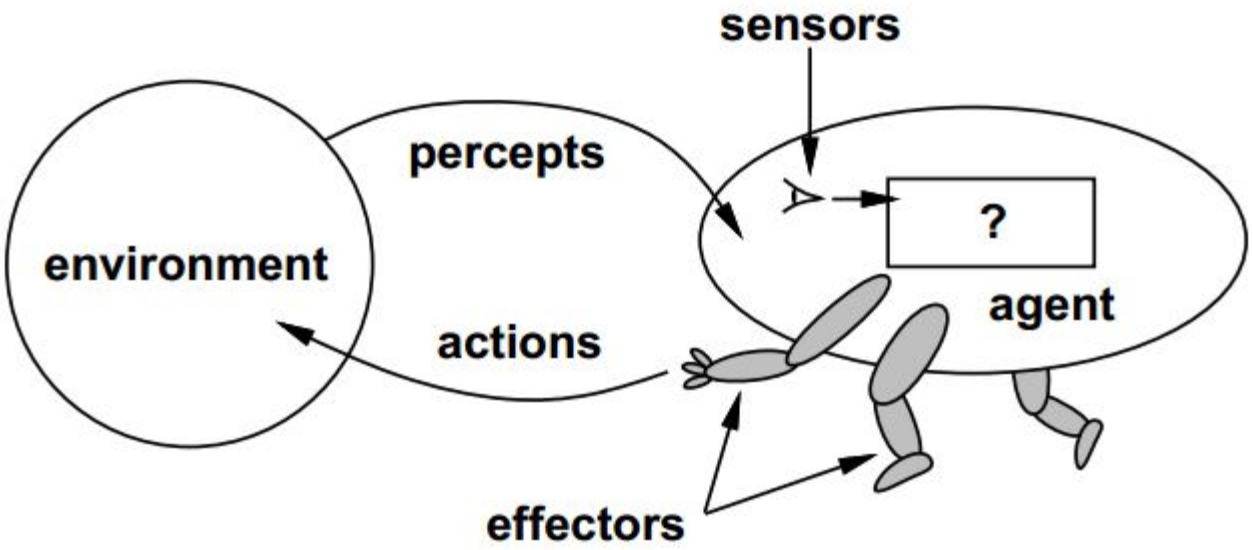
Fig: Agents interact with environments through sensors and actuators

- Examples: Humans and animals, robots and software agents (softbots), temperature control, . . .

Agent and Environments

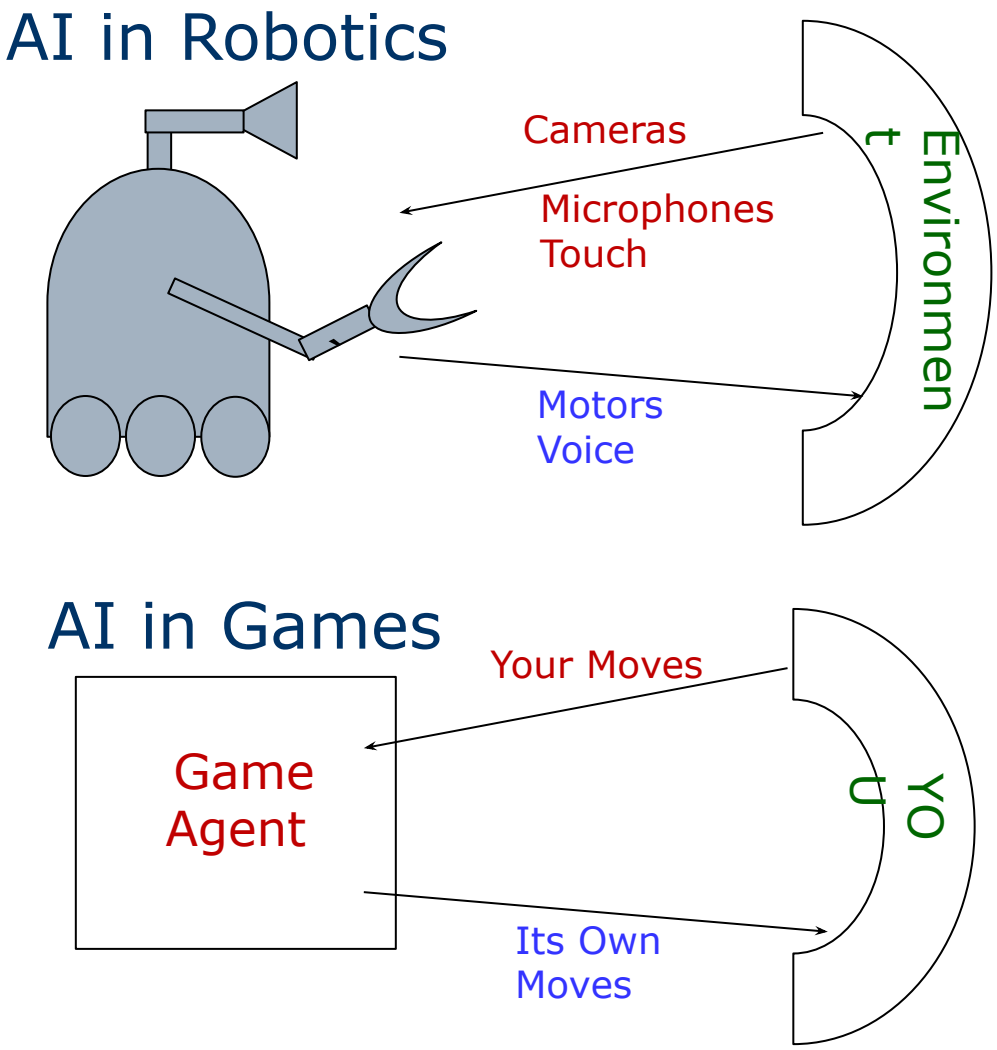
- Agents and environments
  - An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors.

Generic Agent



Agents Interact with environments through sensors and Actuators/effectors

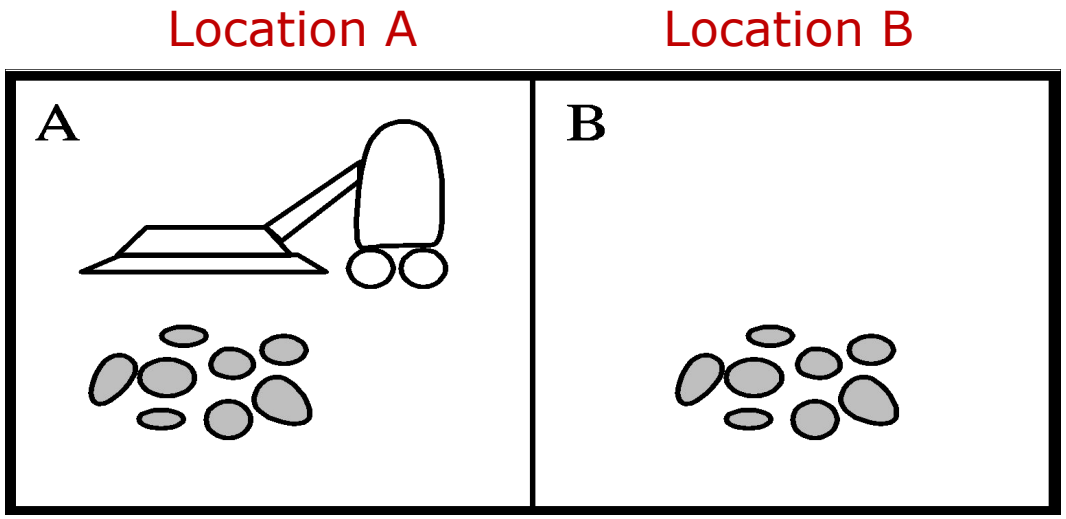
Intelligent Agent - Example



Example: Vacuum Cleaner Agent

- Agent: Robot vacuum cleaner
- Environment: Floors of the house
- Sensors:
  - Dirt sensor: detects when floor in front of robot is dirty
  - Bump sensor: detects when it has bumped into something
  - Power sensor: measures amount of power in battery
  - Bag sensor: amount of space remaining in dirt bag
- Actuators/Effectors:
  - Motorized wheels
  - Suction motor
- Percepts: Location and contents, e.g., [A;Dirty]
- Actions: Left, Right, Pick the Dust, No Op

Example – Vacuum Cleaner world

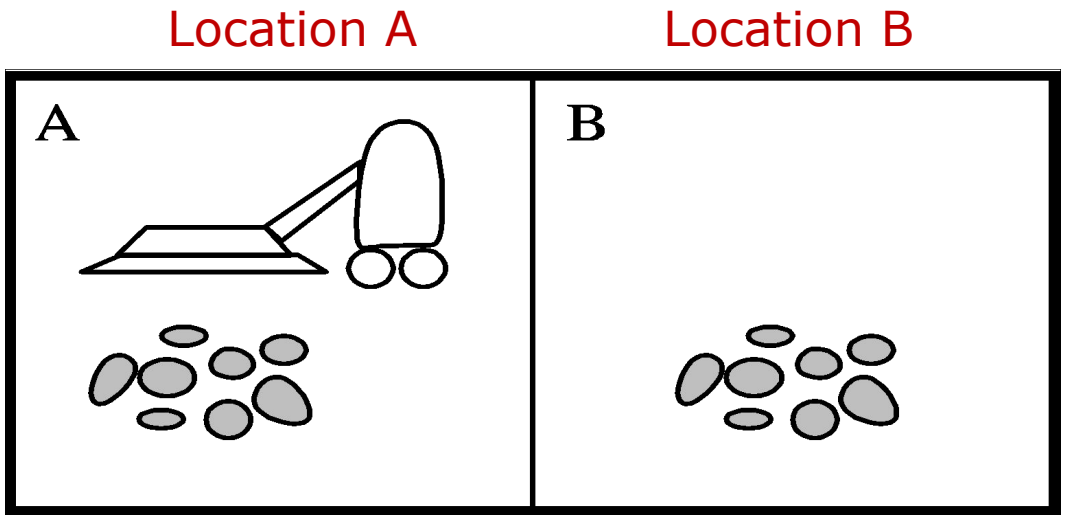


A vacuum-cleaner world with two locations

Partial Tabulation of Agent Function

Percepts Sequence	Action
[A, Clean]	
[A, Dirty]	
[B, Clean]	
[B, Dirty]	

Example – Vacuum Cleaner world



A vacuum-cleaner world with two locations

Partial Tabulation of Agent Function

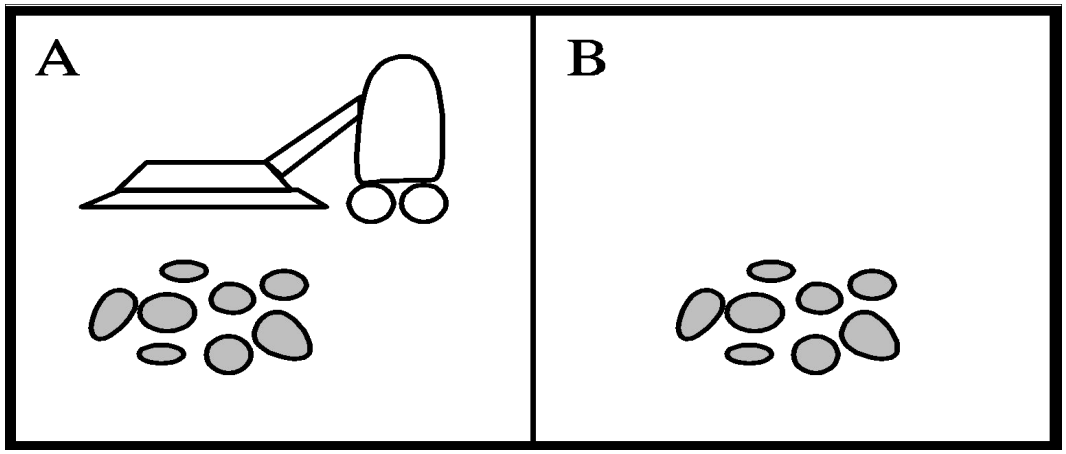
Percepts Sequence	Action
[A, Clean]	Right
[A, Dirty]	Pick the Dust
[B, Clean]	Left
[B, Dirty]	Pick the Dust

Example – Vacuum Cleaner world

Agent Program

Function REFLEX-VACUUM-AGENT([*location, status*]) returns an action

If *status=Dirty* then return *Pick the Dirt*
else if *location = A* then return *Right*
else if *location = B* then return *Left*



Question

An AI agent perceives and acts upon the environment using___.

1. Sensors
2. Perceiver
3. Actuators
4. Both 1 and 3

Question

An AI agent perceives and acts upon the environment using___.

1. Sensors
2. Perceiver
3. Actuators
4. **Both 1 and 3**

Question

If a robot is able to change its own trajectory as per the external conditions, then the robot is considered as the__

1. Mobile
2. Non-Servo
3. Open Loop
4. Intelligent

Question

If a robot is able to change its own trajectory as per the external conditions, then the robot is considered as the__

1. Mobile
2. Non-Servo
3. Open Loop
4. **Intelligent**

Question

What is an ‘agent’?
a) Perceives its environment through sensors and acting upon that environment through actuators
b) Takes input from the surroundings and uses its intelligence and performs the desired operations
c) A embedded program controlling line following robot
d) All of the mentioned

Question

What is an ‘agent’?
a) Perceives its environment through sensors and acting upon that environment through actuators
b) Takes input from the surroundings and uses its intelligence and performs the desired operations
c) A embedded program controlling line following robot
d) **All of the mentioned**

Question

Agents behavior can be best described by

_____

a) Perception sequence
b) Agent function
c) Sensors and Actuators
d) Environment in which agent is performing

Question

Agents behavior can be best described by _____

a) Perception sequence

b) **Agent function**

c) Sensors and Actuators

d) Environment in which agent is performing

Explanation: An agent's behavior is described by the agent function that maps any given percept sequence to an action, which can be implemented by agent program. The agent function is an abstract mathematical description; the agent program is a concrete implementation, running on the agent architecture.

The main tasks of an AI agent are_____.

- Input and Output
- Moment and Humanly Actions
- Perceiving, thinking, and acting on the environment
- None of the above

Question

The main tasks of an AI agent are_____.

- Input and Output
- Moment and Humanly Actions
- **Perceiving, thinking, and acting on the environment**
- None of the above

Concept of Rationality

Rational Agents

. . . do the "right thing"!

In order to evaluate their performance, we have to define a performance measure.

Autonomous vacuum cleaner example:

- m^2 per hour
- Level of cleanliness
- Energy usage
- Noise level
- Safety

Optimal behavior is often unattainable!

- Not all relevant information is perceivable
- Complexity of the problem is too high

The Ideal Rational Agent

Rational behavior depends on:

- Performance measures (goals)
- Percept sequences
- Knowledge of the environment
- Possible actions

Ideal rational agent

For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Percept Sequence × World Knowledge → Action

Rationality: omniscience, learning, and autonomy

- An **omniscient** (**perfect**) agent knows the actual outcome of its actions and can act accordingly; but perfection is impossible in reality.
- **Rationality** is NOT the same as **perfection**.
  - **Rationality** maximizes *expected performance*, while **perfection** maximizes *actual performance*.
- A rational agent not only to **gather information (exploration)** but also to **learn** as much as possible from what it perceives.
- An agent relies on the prior knowledge of its designer rather than on its own percepts, we say that the agent lacks **autonomy**. A rational agent should be **autonomous.**

- **Rational ➔ exploration, learning, autonomy**

Question

Rational agent always does the right things.

1. True
2. False

Question

Rational agent always does the right things.

1. **True**
2. False

Question

What is rational at any given time depends on

a) The performance measure that defines the criterion of success
b) The agent's prior knowledge of the environment
c) The actions that the agent can perform
d) All of the mentioned

Question

What is rational at any given time depends on

a) The performance measure that defines the criterion of success
b) The agent's prior knowledge of the environment
c) The actions that the agent can perform
d) **All of the mentioned**

Nature of the Environment

Nature of Environment

Specifying Task Environment:

To design a rational agent, we must specify the task environment

The **Performance Measure, the environment, and the agents actuators and sensors** are grouped as the **task Environment** and called as PEAS description.

Consider, e.g., the task of designing an automated taxi:

- **P**erformance measure: Safety, destination, profits, legality, comfort, ……
- **E**nvironment: Indian streets/freeways, traffic, pedestrians, weather, ……
- **A**ctuators: Steering, accelerator, brake, horn, speaker/display, ……
- **S**ensors: Video, accelerometers, gauges, engine sensors, keyboard, GPS, ……

Task Environment: PEAS

- To design *a rational agent*, we must specify the **task environment**.
- The performance measure, the environment, and the agent's actuators and sensors are grouped as the **task environment**, and called as **PEAS** (Performance measure, Environment, Actuators, Sensors).

Agent Type	Performance Measure	Environment	Actuators	Sensors
Taxi driver	Safe, fast, legal, comfortable trip, maximize profits	Roads, other traffic, pedestrians, customers	Steering, accelerator, brake, signal, horn, display	Cameras, sonar, speedometer, GPS, odometer, accelerometer, engine sensors, keyboard

Figure PEAS description of the task environment for an automated taxi.

Nature of Environment

Specifying Task Environment:

Example: Internet Shopping Agent

PEAS Description

- Performance measure: Price, quality, appropriateness, efficiency……
- Environment: Current Website, Customers, Shippers……
- Actuators: Display to user, follow URL, fill in form……
- Sensors: HTML pages (text, graphics, scripts)

AI Agents with PEAS Examples

Agent	Performance Measure	Environment	Actuator	Sensor
Hospital Management System	Patient's health, Admission process, Payment	Hospital, Doctors, Patients	Prescription, Diagnosis, Scan report	Symptoms, Patient's response
Automated Car Drive	The comfortable trip, Safety, Maximum Distance	Roads, Traffic, Vehicles	Steering wheel, Accelerator, Brake, Mirror	Camera, GPS, Odometer
Subject Tutoring	Maximize scores, Improvement is students	Classroom, Desk, Chair, Board, Staff, Students	Smart displays, Corrections	Eyes, Ears, Notebooks
Part-picking robot	Percentage of parts in correct bins	Conveyor belt with parts; bins	Jointed arms and hand	Camera, joint angle sensors

Question

For each of the following activities, give a PEAS description of the task environment

- Playing soccer.
- Exploring the subsurface oceans of Titan.
- Shopping for used AI books on the Internet.
- Playing a tennis match.
- Practicing tennis against a wall.
- Performing a high jump.
- Knitting a sweater.
- Bidding on an item at an auction

Agent	Performance Measure	Environment	Actuators	Sensors
Playing Soccer	Score, Injuries, Team work.	Players, Referees, Field, Crowd, Goals, Ball.	Strength, Stamina, Coordination.	Eyes, Ears, Mouth, Ears, Touch.
Exploring Titan	Underwater mobility, Safety, Data, Navigation.	Shuttle, Rover, Atmosphere, Surface, Ocean.	Communication, Sustainability, Reliability.	Camera, GPS, Temperature, Pressure.
AI Book Shopping	Prices, Ease of site, Shipping time.	Websites, Internet, PC.	Correct Information, User.	Pictures, Information, Eyes.
Playing Tennis	Scoring, Stamina, Team work, Strategy.	Players, Referees, Crowd, Net, Court, Ball.	Strength, Stamina, Coordination.	Eyes, Skill, Footwork.
Practicing Tennis	Stamina, Lowering missed balls.	Player, Wall, Racket, Ball.	Stance, Racket Placement, Speed.	Eyes, Skill, Footwork.
High Jump	Form, Height, Landing.	Height bar, Padded Mat, Judge, Field.	Speed, Form, Leg Strength, Flexibility.	Eyes, Touch.
Knitting a Sweater	Correct Dimension, Reducing mistakes.	Yarn, Needles, Instructions, Room.	Speed, Yarn type, Sweater size, Precision.	Eyes, Hands.
Auction Bidding	Winning, Paying lowest price.	Opponents, Item, Auctioneer.	Budget, Item Value, Eagerness.	Eyes, Ears, Mouth, Knowledge of item.

Properties of task environments

An environment in artificial intelligence is the surrounding of the agent. The agent takes input from the environment through sensors and delivers the output to the environment through actuators.

There are several types of environments:

1. Fully Observable vs Partially Observable
2. Single-agent vs Multi-agent
3. Competitive vs Co-operative/Collaborative
4. Deterministic vs Non-deterministic/Stochastic
5. Episodic vs Sequential
6. Static vs Dynamic
7. Discrete vs Continuous
8. Known vs Unknown

1. Fully Observable vs Partially Observable

- When an agent sensor is capable to sense or access the complete state of an agent at each point in time, it is said to be a fully observable environment else it is partially observable.
- Maintaining a fully observable environment is easy as there is no need to keep track of the history of the surrounding.
- An environment is called **unobservable/partially** when the agent has no sensors in all environments.
- **Examples:**
  - **Chess –** the board is fully observable, and so are the opponent's moves.
  - **Driving –** the environment is partially observable because what's around the corner is not known.

2. Single-agent vs Multi-agent

- An environment consisting of only one agent is said to be a **single-agent** environment.
  - Example: A person left alone in a maze is an example of the single-agent system.
- An environment involving more than one agent is a **multi-agent** environment.
  - Example:The game of football is multi-agent as it involves 11 players in each team.

3. Competitive vs Co-operative/Collaborative

- An agent is said to be in a competitive environment when it competes against another agent to optimize the output.
  - Example: The game of chess is competitive as the agents compete with each other to win the game which is the output.
- An agent is said to be in a Co-operative/collaborative environment when multiple agents cooperate to produce the desired output.
  - Example: When **multiple self-driving cars** are found on the roads, they **cooperate** with each other to avoid collisions and reach their destination which is the output desired.

4. Deterministic vs Stochastic

- When a uniqueness in the agent's current state completely determines the next state of the agent, the environment is said to be **deterministic**.
- The **stochastic** environment is random in nature which is not unique and cannot be completely determined by the agent.
- **Examples:**
  - Deterministic **Chess –** there would be only a few possible moves for a chess piece at the current state and these moves can be determined.
  - Stochastic **Self-Driving Cars-** the actions of a self-driving car are not unique, it varies time to time.

5. Episodic vs Sequential

- In **an Episodic task environment**, each of the agent's actions is divided into atomic incidents or episodes. There is *no dependency between current and previous incidents*. In each incident, an agent receives input from the environment and then performs the corresponding action.
- **Example:**
  - Consider an example of **Pick and Place robot**, which is used to detect defective parts from the conveyor belts. Here, every time robot(agent) will make the decision on the current part i.e. there is no dependency between current and previous decisions.
- In a **Sequential environment**, the previous decisions can affect all future decisions. The next action of the agent depends on what action he has taken previously and what action he is supposed to take in the future.
- **Example:**
  - **Checkers-** Where the previous move can affect all the following moves.

6. Static vs Dynamic

- An environment that keeps constantly changing itself when the agent is up with some action is said to be dynamic.
  - Example: A roller coaster ride is dynamic as it is set in motion and the environment keeps changing every instant.
- An idle environment with no change in its state is called a static environment.
  - Example: An empty house is static as there's no change in the surroundings when an agent enters.

7. Discrete vs Continuous

- If an environment consists of a **finite number of actions** that can be deliberated in the environment to obtain the output, it is said to be a discrete environment.
  - Example: The game of chess is discrete as it has only a finite number of moves. The number of moves might vary with every game, but still, it's finite.
- The environment in which the actions are performed cannot be numbered i.e., is not discrete, is said to be continuous.
  - Example: Self-driving cars are an example of continuous environments as their actions are *driving, parking*, etc. which cannot be numbered.

8. Known vs Unknown

- In a known environment, the *results* for all actions are known to the agent.
  - Example: Card games
- In unknown environment, the agent needs to learn how it works in order to perform an action.
  - Example: New Video games

Examples of Task Environment

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword Puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Driving Robot car						

Examples of Task Environment

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword Puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Driving Robot car	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi

Properties of task environments

Fully Observable / Partially observable: If it is possible in principle to determine the complete state of the environment at each time point it is observable, Example: Chess game; otherwise it is only partially observable, Example: Driving

Single agent / Multiple agents: the environment may contain other agents which may be of the same kind as the agent, or of different kinds, Multi-agent example: Football

Competitive vs Co-operative/Collaborative: An agent is said to be in a competitive environment when it competes against another agent to optimize the output. Example: Chess. An agent is said to be in a Co-operative/collaborative environment when multiple agents cooperate to produce the desired output. Example: Self-Driving

Deterministic / nondeterministic: If the future state of the environment can be predicted in principle given the current state and the set of actions which can be performed it is deterministic, Example: Chess; otherwise it is nondeterministic, Example: Self-Driving

Episodic / sequential: In an episodic task environment, the agent's experience is divided into atomic episodes. In each episode the agent receives a percept and then performs a single action. The next episode does not depend on the actions taken in previous episodes. Example: Pick and Place Robot. In sequential environments, on the other hand, the current decision could affect all future decisions. Example: Chess game

Static / dynamic: An idle environment with no change in its state is called a static environment.

, Example: Empty house; An environment that keeps constantly changing itself when the agent is up with some action is said to be dynamic.

, Example: Rollar Coaster .

Discrete / continuous: If there are a limited number of distinct, clearly defined, states of the environment, the environment is discrete, Example: Chess board. ; otherwise it is continuous, Example: Diving

Known/ unknown: In a known environment, the outcomes for all actions are given. Example: Card game. If the environment is unknown, the agent will have to learn how it works in order to make good decisions. Example: Video game

Question

For each of the following activities, give a PEAS description of the task environment

- Playing soccer.
- Exploring the subsurface oceans of Titan.
- Shopping for used AI books on the Internet.
- Playing a tennis match.
- Practicing tennis against a wall.
- Performing a high jump.
- Knitting a sweater.
- Bidding on an item at an auction

Agent	Performance Measure	Environment	Actuators	Sensors
Playing Soccer	Score, Injuries, Team work.	Players, Referees, Field, Crowd, Goals, Ball.	Strength, Stamina, Coordination.	Eyes, Ears, Mouth, Ears, Touch.
Exploring Titan	Underwater mobility, Safety, Data, Navigation.	Shuttle, Rover, Atmosphere, Surface, Ocean.	Communication, Sustainability, Reliability.	Camera, GPS, Temperature, Pressure.
AI Book Shopping	Prices, Ease of site, Shipping time.	Websites, Internet, PC.	Correct Information, User.	Pictures, Information, Eyes.
Playing Tennis	Scoring, Stamina, Team work, Strategy.	Players, Referees, Crowd, Net, Court, Ball.	Strength, Stamina, Coordination.	Eyes, Skill, Footwork.
Practicing Tennis	Stamina, Lowering missed balls.	Player, Wall, Racket, Ball.	Stance, Racket Placement, Speed.	Eyes, Skill, Footwork.
High Jump	Form, Height, Landing.	Height bar, Padded Mat, Judge, Field.	Speed, Form, Leg Strength, Flexibility.	Eyes, Touch.
Knitting a Sweater	Correct Dimension, Reducing mistakes.	Yarn, Needles, Instructions, Room.	Speed, Yarn type, Sweater size, Precision.	Eyes, Hands.
Auction Bidding	Winning, Paying lowest price.	Opponents, Item, Auctioneer.	Budget, Item Value, Eagerness.	Eyes, Ears, Mouth, Knowledge of item.

Characteristic (properties) of the task environment						
Task environment	**Observable**	**Deterministic**	**Episodic**	**Static**	**Discrete**	**Agents**
Playing soccer.	Partially	stochastic	sequential	dynamic	continuous	multi-agent
Shopping for AI books on the Internet.	Partially	deterministic	sequential	Static*	discrete	single agent
Playing a tennis match.	Fully	stochastic	episodic	dynamic	continuous	multi-agent
Practicing tennis against a wall.	Fully	stochastic	episodic	dynamic	continuous	single agent
Mathematician's theorem-proving assistant	Fully	deterministic	sequential	static	discrete	single agent
Autonomous Mars rover	Partially	stochastic	sequential	dynamic	continuous	single-agent

Properties of task environments

1	Fully observable vs Partially observable	Ex. Cross-word Puzzle Vacuum Cleaner
2	Deterministic vs. Stochastic	Ex. Cross-word Puzzle Tossing a coin
3	Episodic vs. sequential	Ex. Identifying Defective part Chess
4	Static vs. Dynamic	Ex. Cross Word Puzzle Taxi Driving
5	Discrete vs. Continuous	Ex. Chess Play Taxi Driving
6	Single agent vs. Multi agent	Ex. Cross-word puzzle Chess play – Two Agent

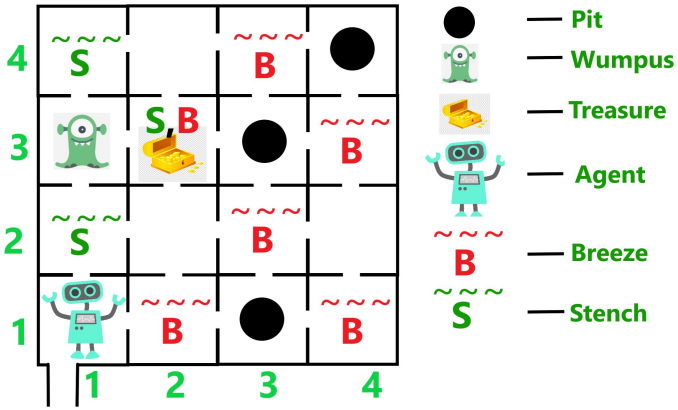
Question

Problem Statement:

The Wumpus world is a cave with *16* rooms *(4×4)*. Each room is connected to others through walkways (no rooms are connected diagonally). The knowledge-based agent starts from *Room[1, 1]*. The cave has – some **pits**, a **treasure** and a beast named **Wumpus**. The Wumpus can not move but eats the one who enters its room. If the agent enters the pit, it gets stuck there. The goal of the agent is to take the treasure and come out of the cave. The agent is rewarded, when the goal conditions are met. The agent is penalized, when it falls into a pit or being eaten by the Wumpus.

Some elements support the agent to explore the cave, like -The wumpus's adjacent rooms are stenchy. -The agent is given one arrow which it can use to kill the wumpus when facing it (Wumpus screams when it is killed). – The adjacent rooms of the room with pits are filled with breeze. -The treasure room is always glittery.

Write the PEAS description and properties of the task environment



Answer

1. **PEAS Description** for the Wumpus World problem:**Performance measures:**
   1. Agent gets the gold and return back safe = *+1000 points*
   2. Agent dies = *-1000 points*
   3. Each move of the agent = *-1 point*
   4. Agent uses the arrow = *-10 points*
2. **Environment:**
   1. A cave with *16(4×4)* rooms
   2. Rooms adjacent (not diagonally) to the Wumpus are stinking
   3. Rooms adjacent (not diagonally) to the pit are breezy
   4. The room with the gold glitters
   5. Agent's initial position – *Room[1, 1]* and facing right side
   6. Location of Wumpus, gold and *3 pits* can be anywhere, except in *Room[1, 1]*.
3. **Actuators:**
   Devices that allow the agent to perform the following actions in the environment.
   1. Move forward
   2. Turn right
   3. Turn left
   4. Shoot
   5. Grab
   6. Release

4. Sensors: Devices which helps the agent in sensing the following from the environment.

1. Breeze
2. Stench
3. Glitter
4. Scream (When the Wumpus is killed)
5. Bump (when the agent hits a wall)

Wumpus World Characterization:

- **Partially Observable:** knows only the local perceptions
- **Deterministic:** outcome is precisely specified
- **Sequential:** subsequent level of actions performed
- **Static:** Wumpus, pits are immobile
- **Discrete:** discrete environment
- **Single-agent:** The knowledge-based agent is the only agent whereas the wumpus is considered as the environment's feature.

Question

There are __ types of observing environments?

a) 4

b) 3

c) 2

d) 0

Answer with explanation: The correct answer is option is c. there are two types of observing environments and these are Fully and Partial environments.

Question

- Crossword puzzle environment in artificial intelligence

(A). Dynamic

(B). Static

(C). Semi Dynamic

(D). None of these

Answer:B

Structure of Rational Agents

Realization of the ideal mapping through an

- Agent program, executed on an
- Architecture which also provides an interface to the environment (percepts, actions).

Agent = Architecture + Program

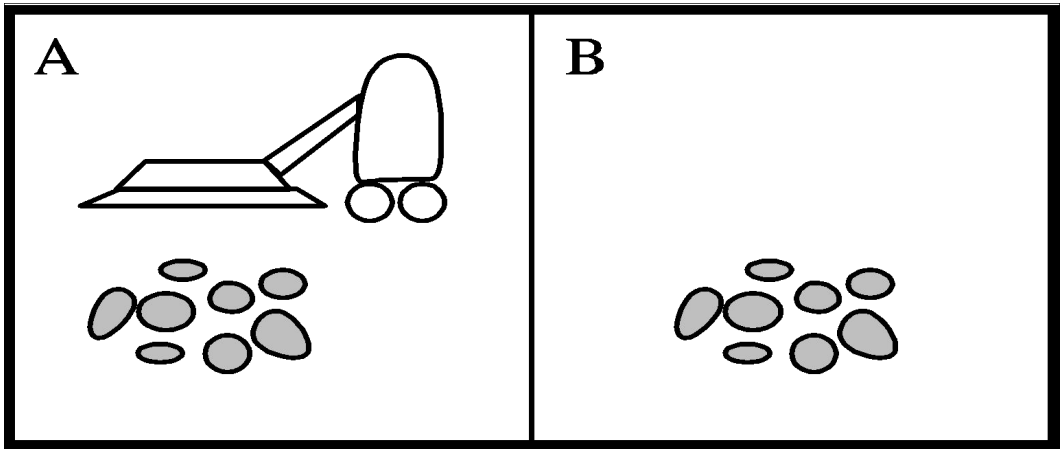
The Simplest Design: Table-Driven Agents

```
function TABLE-DRIVEN-AGENT(percept) returns an action
  persistent: percepts, a sequence, initially empty
              table, a table of actions, indexed by percept sequences, initially fully specified

  append percept to the end of percepts
  action ← LOOKUP(percepts, table)
  return action
```

Example – Vacuum Cleaner world

Location A Location B



A vacuum-cleaner world with two locations

Partial Tabulation of Agent Function

Percepts Sequence	Action
[A, Clean]	Right
[A, Dirty]	Pick the Dust
[B, Clean]	Left
[B, Dirty]	Pick the Dust

The Simplest Design: Table-Driven Agents

Problems:

- The table can become very large.
- It usually takes a very long time for the designer to specify it.
- . . . practically impossible!

Intelligent Agent Types

Four basic types in order of increasing generality:

1. Simple reflex agents
2. Model-based reflex agents
3. Goal-based agents
4. Utility-based agents

Simple reflex agents

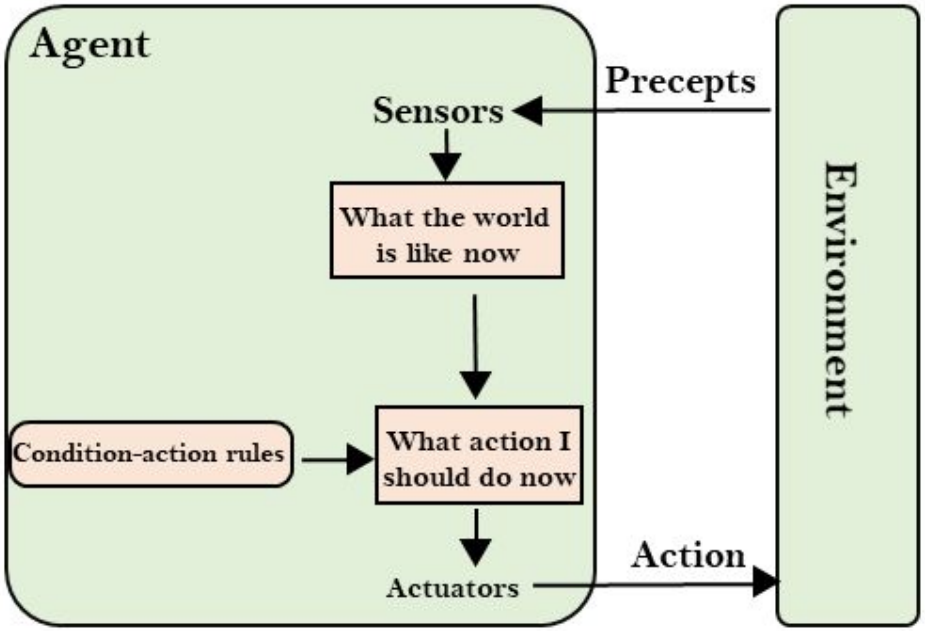
- Act on basis of current perception
- Ignore the rest of the percept history
- Based on If-Then rules
- Environment should be fully observable

Simple reflex agents

Example:

Thermostat	It senses the room temperature and turns the heater on or off based on a pre-set temperature range
Light sensor in a street lamp	It detects darkness and triggers the lamp to turn on
Vending machine	You select a product, and the machine dispenses it based on your button press

Simple reflex agents



Reflex: An action that is performed without conscious thought as a response to a stimulus
Reflex: Immediately or Spontaneously

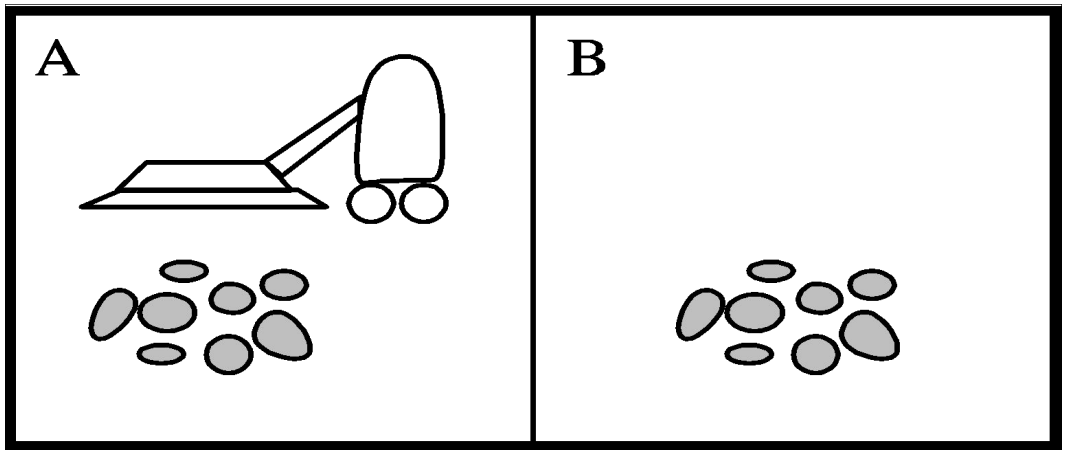
Simple reflex agents

- Example – Vacuum Cleaner world

Agent Program

Function REFLEX-VACUUM-AGENT([*location, status*]) returns an action

If *status=Dirty* then return *Pick the Dirt*
else if *location = A* then return *Right*
else if *location = B* then return *Left*



Simple reflex agents

How they work:

- They operate in a continuous loop of perception and action. Sensors capture information about the environment
- This information is then matched against a set of pre-programmed rules, which are like a massive "IF…THEN…" list.
- Based on the matched rule, the agent takes a pre-defined action

Simple reflex agents

Strengths:

- They are simple and easy to implement
- Simple reflex agents are fast and efficient
- These AI agents are suitable for well-defined environments

Weaknesses:

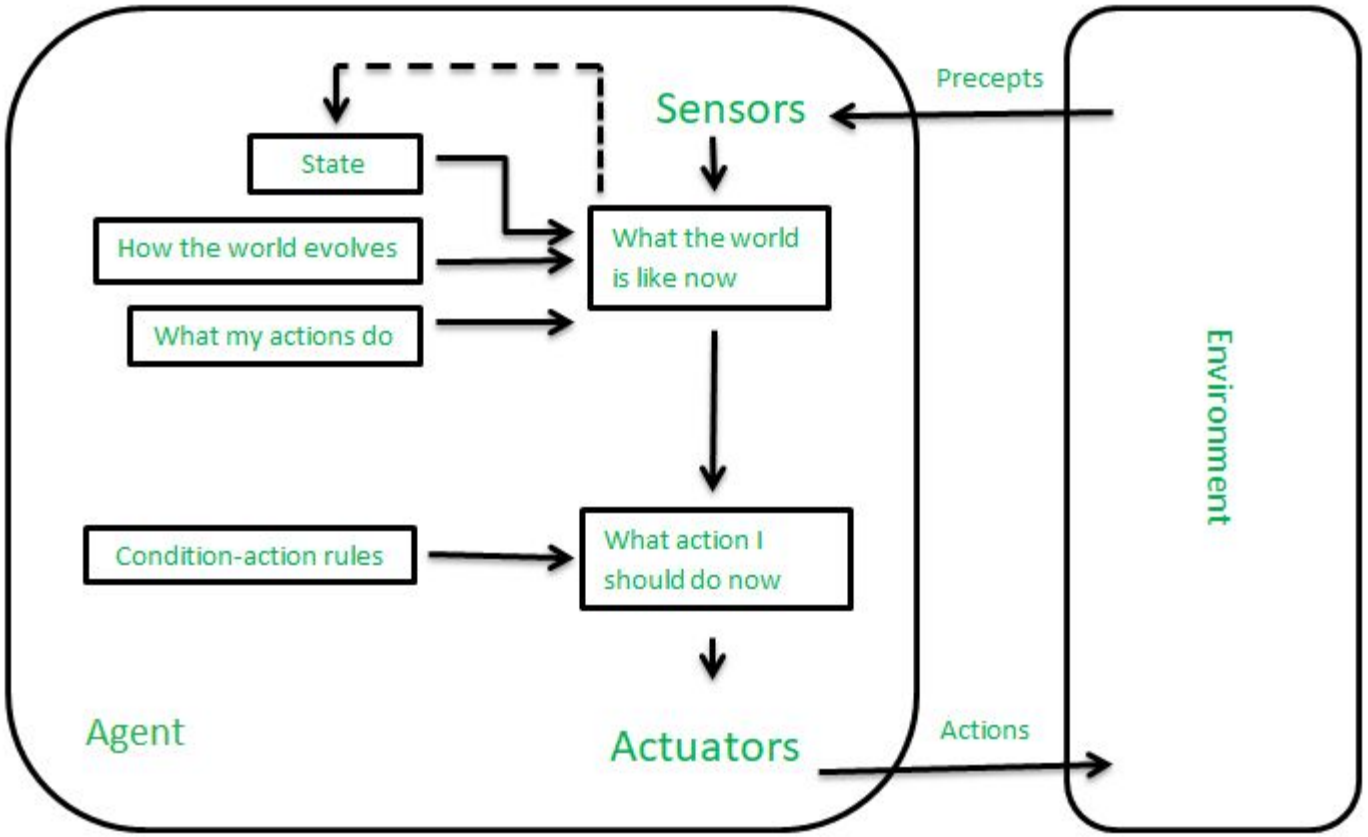
- They have limited adaptability
- They cannot learn from past experiences
- These agents require a fully observable environment
- Lacking history, easily get stuck in infinite loops

Model-based Reflex Agents

- Model-based: Knowledge based.
- Model: Storing Percept History
- The agent has to keep track of the **internal state** which is adjusted by each percept and that depends on the percept **history.**
- A model-based agent can handle **partially observable environments** by the use of a model about the world.
- The current state is stored inside the agent which maintains some kind of structure describing the part of the world which cannot be seen.

Model-based Reflex Agents

- Reflex agents with state



Model-based Reflex Agents

Example:

Self-driving cars	These automobiles rely on internal models of the road network, traffic lights, lanes, and potential obstacles to navigate safely
Chatbots (with context awareness)	A chatbot can maintain an internal model of the conversation to provide more relevant responses

Model-based Reflex Agents

- Reflex agents with state

```
function MODEL-BASED-REFLEX-AGENT(percept) returns an action
  persistent: state, the agent's current conception of the world state
              model, a description of how the next state depends on current state and action
              rules, a set of condition–action rules
              action, the most recent action, initially none

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

Model-based Reflex Agents

How a model-based reflex agent typically operates:

- **Perception**: The agent perceives the current state of the environment through sensors, which provide it with information about the current state, such as the presence of obstacles, objects, or other agents.
- **Modeling the Environment**: The agent maintains an internal model of the environment, which includes information about the state of the world, the possible actions it can take, and the expected outcomes of those actions. This model allows the agent to anticipate the effects of its actions before taking them.
- **Decision Making**: Based on its current perceptual input and its internal model of the environment, the agent selects an action to perform. The selection of actions is typically guided by a set of rules or heuristics that map perceived states to appropriate actions.
- **Action Execution**: The agent executes the selected action in the environment, which may cause changes to the state of the world.
- **Updating the Model**: After taking an action, the agent updates its internal model of the environment based on the new perceptual information it receives. This allows the agent to continuously refine its understanding of the world and improve its decision-making process over time.

Model-based Reflex Agents

Benefits:

- They can handle partially observable environments
- Model-based agents are more flexible
- They can use the internal model to make predictions about how the environment might react to their actions

Drawbacks:

- There is an increased level of complexity.
- The agent's performance relies heavily on the accuracy of its internal model.
- There is always an issue of limited learning. Because, these AI agents rely on pre-programmed rules and don't exhibit true learning capabilities.

Question

Which rule is applied for the Simple reflex agent?

- Simple-action rule
- Simple & Condition-action rule
- Condition-action rule
- None of the above

Question

Which rule is applied for the Simple reflex agent?

- Simple-action rule
- Simple & Condition-action rule
- Condition-action rule
- None of the above

Answer: c. Condition-action rule

- **Explanation:** The simple reflex agent takes decisions only on the current condition and acts accordingly; it ignores the rest of history; hence it follows the Condition-action rule.

Question

Which of these types of intelligent agents relies only on current conditions, making no use of historical data?

- Simple reflex
- Utility-based
- Learning
- Goal-based

Question

Which of these types of intelligent agents relies only on current conditions, making no use of historical data?

- **Simple reflex**
- Utility-based
- Learning
- Goal-based

Question

How does the ''condition-action rule'' work with a simple reflex agent?

- When rules are established, they are based on actions.
- When a condition is met, the agent overrides certain rules.
- When a condition is met, the agent acts based on the rule.
- Conditions mean nothing and rules are made to be broken.

Question

How does the ''condition-action rule'' work with a simple reflex agent?

- When rules are established, they are based on actions.
- When a condition is met, the agent overrides certain rules.
- **When a condition is met, the agent acts based on the rule.**
- Conditions mean nothing and rules are made to be broken.

Question

Model-based agents use which of these things to make decisions about how to act:

- Plastic models
- Internal memory
- External memory
- User entry

Question

Model-based agents use which of these things to make decisions about how to act:

- Plastic models
- **Internal memory**
- External memory
- User entry

Question

What differentiates a model-based reflex agent from a simple reflex agent?

- A model-based agent relies only on current understanding.
- A simple reflex agent is more sophisticated.
- A model-based agent can incorporate percept history.
- A simple reflex agent only looks at percept history.

Question

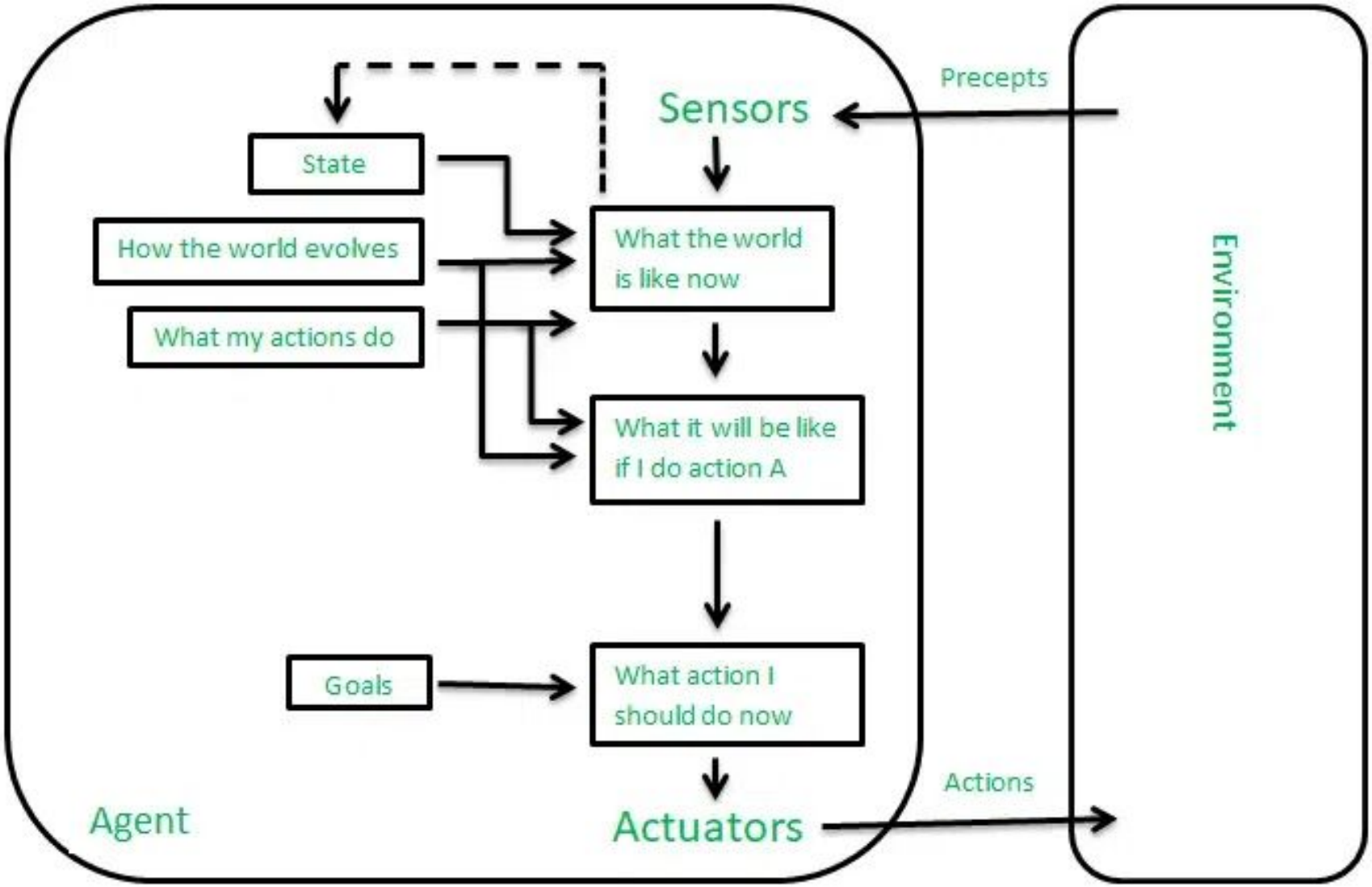
What differentiates a model-based reflex agent from a simple reflex agent?

- A model-based agent relies only on current understanding.
- A simple reflex agent is more sophisticated.
- **A model-based agent can incorporate percept history.**
- A simple reflex agent only looks at percept history.

Goal-based agents

- **Goal-based agents** have predefined objectives or goals that they aim to achieve.
- By combining descriptions of goals and models of the environment, these agents plan to achieve different objectives, like reaching particular destinations.
- They use search and planning methods to create sequences of actions that enhance decision-making in order to achieve goals.
- Goal-based agents differ from reflex agents by including forward-thinking and future-oriented decision-making processes.

Goal-based agents



Goal-based agents

Example:

Robot path planning	A robot might use a goal-based approach to plan its path around obstacles to reach a specific location
Game-playing AI	Chess programs or AI opponents in strategy games employ goal-based decision-making to achieve victory
Navigation apps	These apps use goal-based algorithms to find the best route for you to reach your destination

Goal-based Agents

How they work:

- To reach their goals, these AI agents employ planning algorithms
- The planning process often involves examining a tree of possibilities, with each branch representing a different action the agent can take.
- Goal-based AI agents consider the potential consequences of each action and choose the one that leads it closer to their goal
- They rely on knowledge representation to perform adequate planning. This knowledge base stores information about the environment, the AI agent's capabilities, and the relationships between actions and outcomes

Goal-based Agents

Advantages:

- They can adapt their behavior depending on the current situation
- These AI agents function in environments with multiple possible outcomes
- They have solid reasoning capability

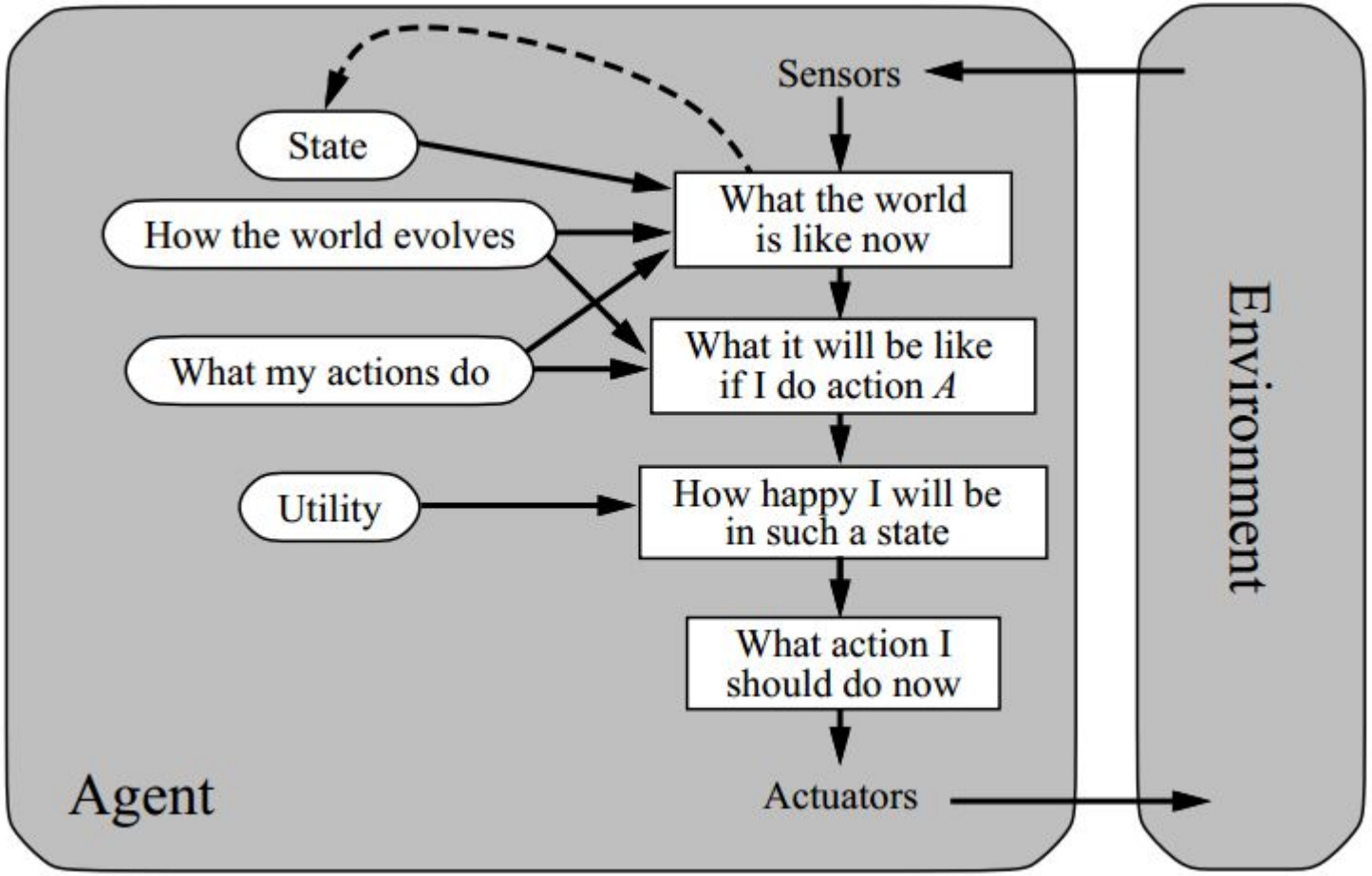
Weaknesses:

- The planning algorithms can be computationally expensive
- Defining clear goals is crucial for the agent's success
- If the agent doesn't have complete information about the environment, its planning might be flawed

Utility-based agents

- When there are multiple possible alternatives, then to decide which one is best, utility-based agents are used.
- They choose actions based on a **preference (utility)** for each state. Sometimes achieving the desired goal is not enough. We may look for a quicker, safer, cheaper trip to reach a destination.
- Agent happiness should be taken into consideration. Utility describes how **"happy"** the agent is. The word "utility" here refers to "the quality of being useful
- Because of the uncertainty in the world, a utility agent chooses the action that maximizes the expected utility. A utility function maps a state onto a real number which describes the associated degree of happiness.

Utility-based agents



Utility-based agents

Example:

Recommendation systems	These systems recommend products, movies, or music to users based on a predicted utility score of how much the user would enjoy them
Self-driving cars	A utility-based self-driving car can consider factors like safety, efficiency, and passenger comfort when making decisions

Utility-based agents

How they work

- Utility-based agents evaluate different courses of action based on a utility function. This function assigns a specific numerical value to each possible outcome, representing how desirable that outcome is for the agent
- The agent strives to maximize its overall score by choosing actions that lead to outcomes with higher utility values
- They gather information about the environment through sensors
- Next, they consider different possible actions to take. For each action, the agent predicts the potential outcomes that can occur
- The utility function assigns a score to each predicted outcome based on how desirable it is for the agent. Then, the agent selects the predicted action to lead to the outcome with the highest utility value

Utility-based agents

Benefits:

- These AI agents are flexible and adaptive
- They can incorporate the agent's preferences and priorities into their decision-making process
- Utility-based agents can consider factors like risk, time, and effort when evaluating different options

Limitations:

- Designing the utility function is complex
- Evaluating the utility of all possible outcomes can be computationally expensive

Learning Agents

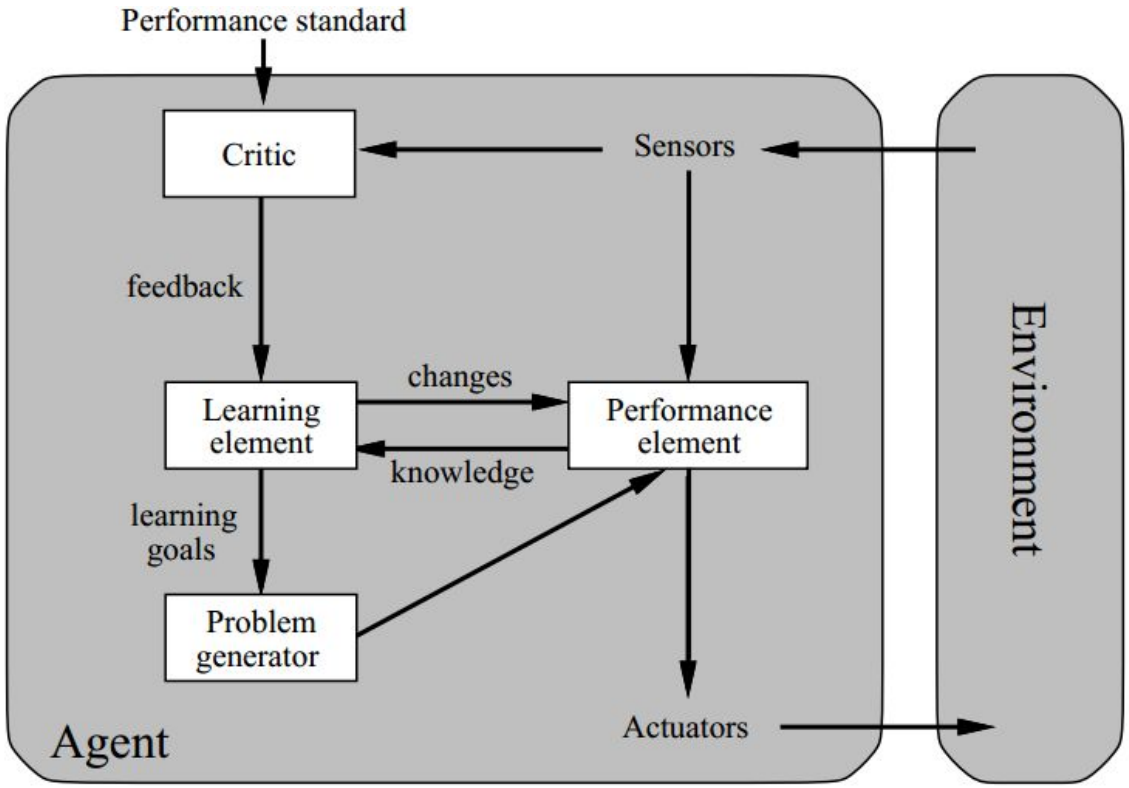
- Learning agents are a key idea in the field of artificial intelligence, with the goal of developing systems that can improve their performance over time through experience. These agents are made up of a few important parts: the learning element, performance element, critic, and problem generator.
- The learning component is responsible for making enhancements based on feedback received from the critic, which evaluates the agent's performance against a fixed standard. This feedback allows the learning aspect to adjust the behavior aspect, which chooses external actions depending on recognized inputs.
- The problem generator suggests actions that may lead to new and informative experiences, encouraging the agent to investigate and possibly unearth improved tactics. Through integrating feedback from critics and exploring new actions suggested by the problem generators, the learning agent can evolve and improve its behavior gradually.

Learning Agents

Examples:

Personal assistants	Virtual assistants like Siri or Alexa learn your preferences and voice patterns to provide more personalized responses
Spam filters	These email filters use machine learning to identify spam emails from past examples
Self-driving cars	These automobiles rely on machine learning to continuously improve their ability to navigate roads and respond to changing situations

Learning Agents



Four Components of Learning Agent

1. Learning element: Responsible for making improvements;

2. Performance element: Responsible for selecting external actions

3. Critic: Gives feedback to the agent, and determines how the performance should be modified;

4. Problem generator: Responsible for suggesting actions that will lead to new and informative experiences

Learning Agents

How learning agents work:

- Learning agents have the potential to learn from their interactions with the environment. This trait allows them to adapt their behavior over time
- These agents have specific components, making them effective. These components are the learning element, critic, performance element, and knowledge representation
- Now, the working of learning agents is a collective result of each component's utility
- The learning element is responsible for processing new information and updating the agent's knowledge or decision-making strategy
- Next, the critic evaluates the agent's performance and provides feedback on how well it's doing compared to its goals
- The performance element selects actions for the agent to take in the environment based on its current knowledge and the critic's feedback
- Finally, knowledge representation refers to how the agent stores and organizes information about the environment and itself

Learning Agents

Benefits:

- Learning agents can adjust to new situations and environments by continuously improving their performance
- They can handle complex tasks
- These AI agents have real-world applicability

Drawbacks:

- Learning agents require a significant amount of data and time
- The agent needs to balance exploring new options for learning with exploiting its current knowledge for good performance
- Understanding how a learning agent arrived at a particular decision can be challenging

Summary

Simple Reflex agent : take decisions on the basis of the current environment and ignore the history.

Model-based reflex agent : work partially in the observable environment, and track the situation.

Goal-based agents : needs to know its goal that describes advisable situations.

Utility-based agents : acts based not only goal but also the best possible way to achieve it.

Learning Agents : learn from its past experiences, or it has learning capabilities.

Question

Which agent deals with the happy and unhappy state?

- ☐ Utility-based agent
- ☐ Model-based agent
- ☐ Goal-based Agent
- ☐ Learning Agent

Question

Which agent deals with the happy and unhappy state?

- Utility-based agent
- Model-based agent
- Goal-based Agent
- Learning Agent
- **Answer:** Utility-based agent
- **Explanation:** Utility-based agent uses an extra component of utility that provides a measure of success at a given state. It decides that how efficient that state to achieve the goal, which specifies the happiness of the agent.

Questionnaire

What kind of agent are you?

- Table-Driven
- Utility-Based
- Reflex Agent
- Learning

Question

What does a goal-based agent do that a model-based agent doesn't?

- It works toward a specific outcome.
- It assesses its current environment.
- It predicts future behaviors or outcomes.
- It relies only on memory percepts.

Question

What does a goal-based agent do that a model-based agent doesn't?

- **It works toward a specific outcome.**
- It assesses its current environment.
- It predicts future behaviors or outcomes.
- It relies only on memory percepts.

Question

How does a learning agent ''learn''?

- By being programmed with all possible solutions and outcomes.
- By gathering feedback and finding new experiences for an action.
- By relying on human intervention to foresee new experiences.
- By forgetting past outcomes and focusing on future plans.

Question

How does a learning agent ''learn''?

- By being programmed with all possible solutions and outcomes.
- **By gathering feedback and finding new experiences for an action.**
- By relying on human intervention to foresee new experiences.
- By forgetting past outcomes and focusing on future plans.

Question

Which component of a learning agent is responsible for gathering feedback?

- The critic element
- The performance element
- The problem generator
- The learning element

Question

Which component of a learning agent is responsible for gathering feedback?

- **The critic element**
- The performance element
- The problem generator
- The learning element

Question

- What role does the problem generator play in a learning agent?
- It delivers feedback suggestions.
- It performs initial operations.
- It suggests new experiences.
- It questions agent responsibilities.

Question

- What role does the problem generator play in a learning agent?
- It delivers feedback suggestions.
- It performs initial operations.
- **It suggests new experiences.**
- It questions agent responsibilities.

Unit-1

Unit-1: Definition, Agents: Agents and environment, Concept of Rationality, The nature of environment, The structure of agents. Problem-solving: Problem-solving agents, Example problems, Searching for Solutions.

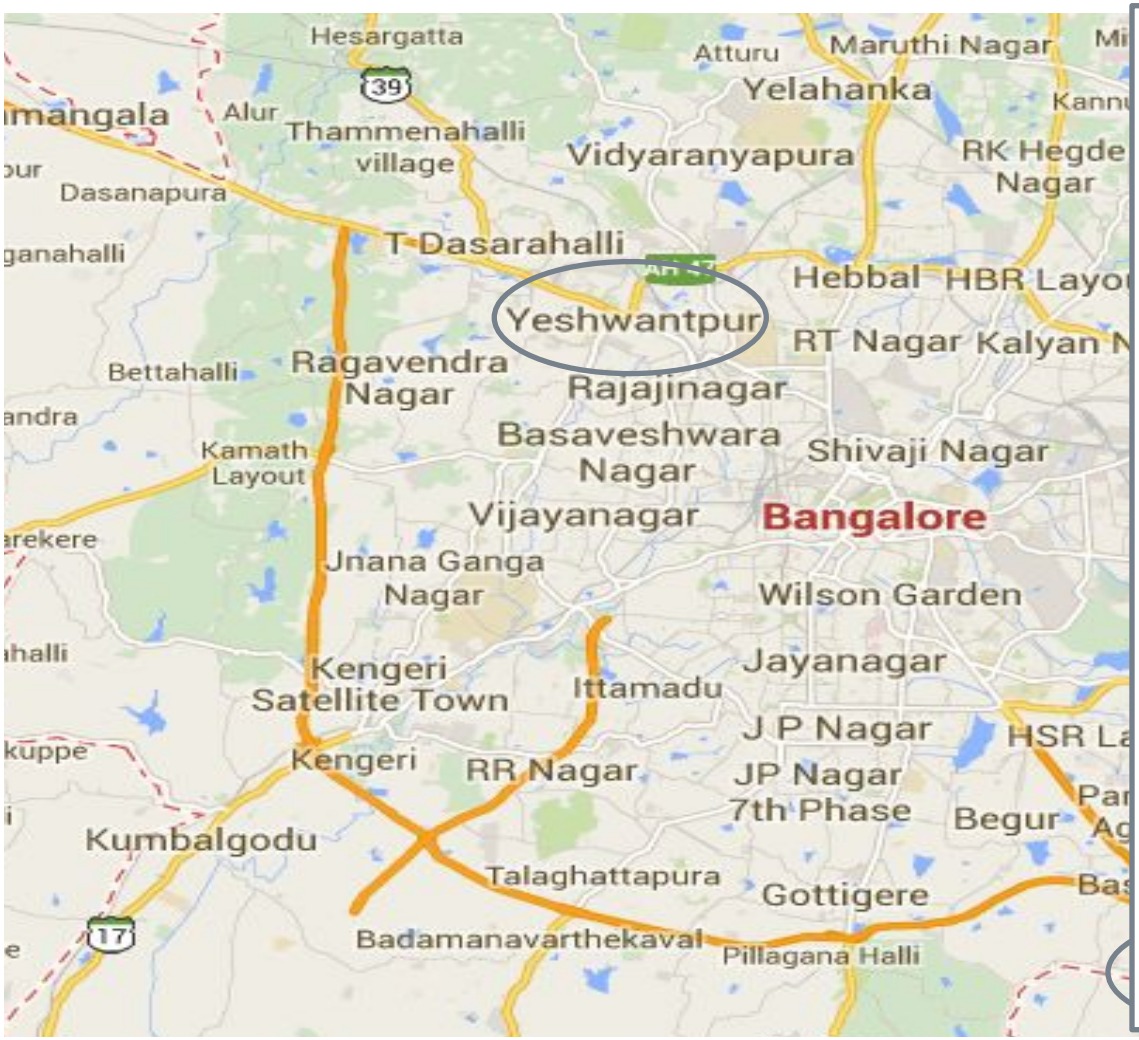
Problem Solving - Definition

- Problem solving is a process of generating solutions from observed data.
- Problem is characterized by a set of goals, set of objects, and set of operations.
- The method of solving problem through AI involves the process of
  - defining the search space,
  - deciding start and goal states
  - finding the path from start state to goal state through search space.

Problem Solving - Definition

- Here search space or problem space is abstract space which has all valid states that can be generated by the application of any combination of operators or objects.
- A problem space can have one or more solutions.
- Solution is a combination of objects and operations that achieve the goals.
- Search refers a search of solution in problem space.

Problem Solving – Route Map

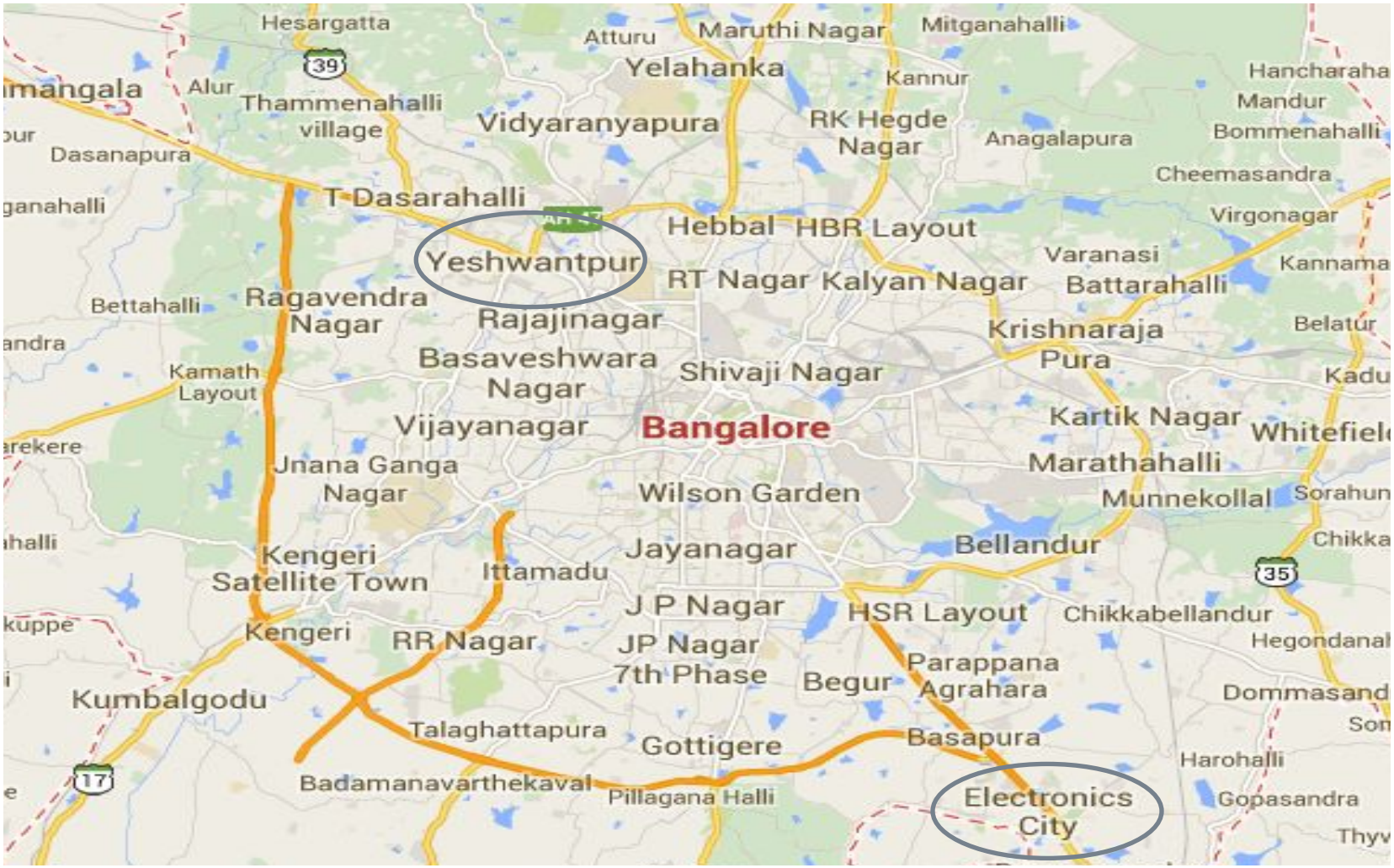


Find route map from Yeshwantpur to Electronics City?

Can the **agent** come up with solution given this route map

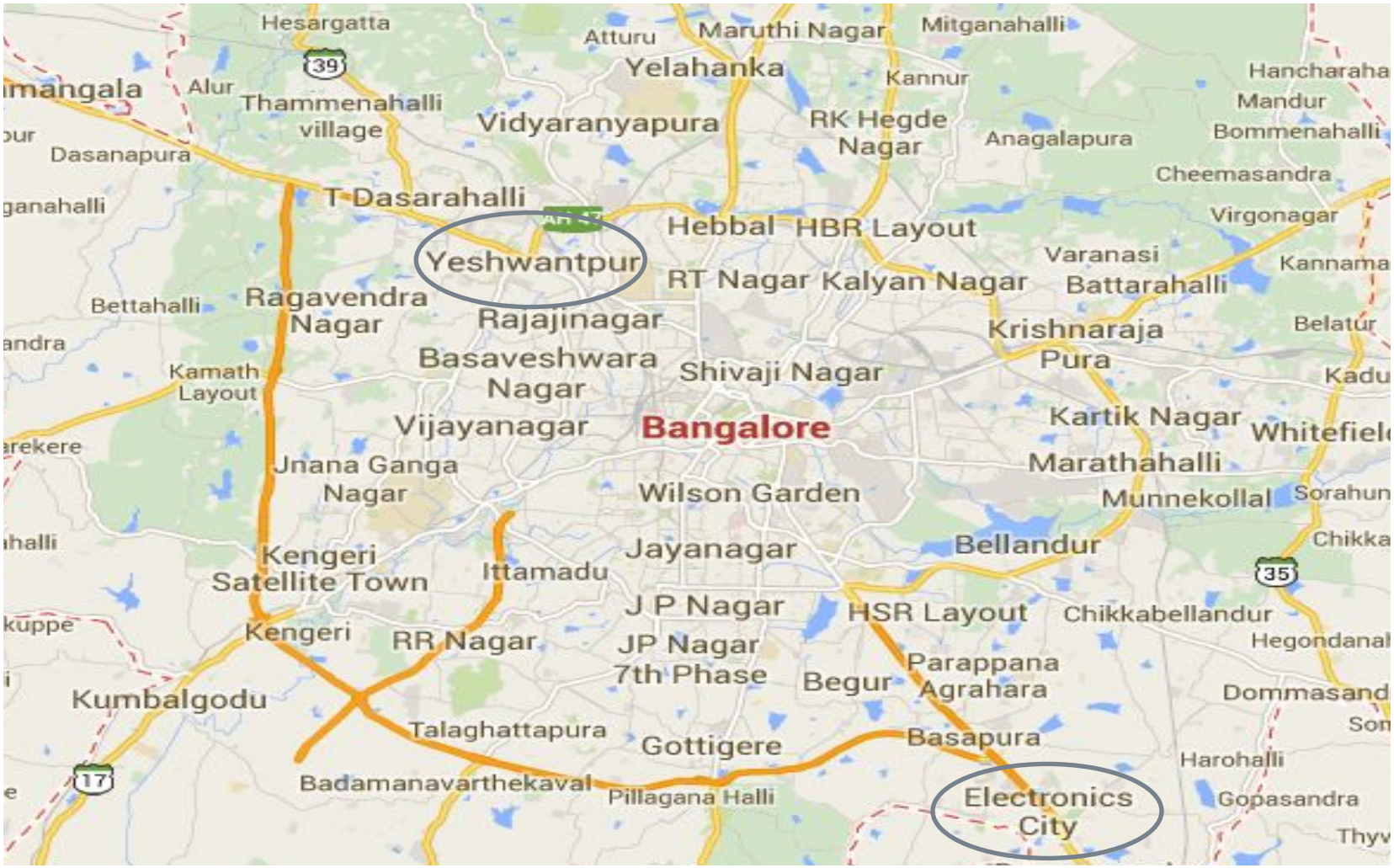
- ❑ Yes
- ❑ No

Problem Solving – Route Map



Problem Solving – Route Map

Problem Solving by Search



Problem Solving Agent or Goal Based Agent

- Goal-based agents are usually called planning agents.
- Problem solving begins with the definitions of problems and their solutions.
- We then describe several general-purpose search algorithms that can be used to solve these problems.
- We will see several uninformed search algorithms— algorithms that are given no information about the problem other than its definition.
- Informed search algorithms, on the other hand, can do quite well given some guidance on where to look for solutions

Problem-Solving Agent

- A **problem-solving agent** is a *goal-based agent* and use *atomic representations.*
  - In atomic representations, states of the world are considered as wholes, with no internal structure visible to the problem solving algorithms.
- *Intelligent agents* are supposed to maximize their *performance measure*. Achieving this is sometimes simplified if the agent can adopt a **goal** and aim at satisfying it.
- **Problem formulation** is the process of deciding what actions and states to consider, given a *goal*.
- The process of looking for a sequence of actions that reaches the goal is called **search**.
- A ***search algorithm*** takes a problem as input and returns a **solution** in the form of an action sequence.
- Once a *solution* is found, the carrying actions it recommends is called the **execution phase**.

- A problem-solving agent has three phases:
  - **problem formulation, searching solution and executing actions in the solution.**

Problem-Solving Agent: Well-defined problems

A problem can be defined by five components:

1. Initial state
2. Actions/Operators/Successor Function
3. Transition model or State Space
4. Goal test
5. Path cost

Problem-Solving Agent: Well-defined problems

A **problem** can be defined by five components:

- **initial state, actions, transition model, goal test, path cost.**

INITIAL STATE: The **initial state** that the agent starts in.

ACTIONS: A description of the possible **actions** available to the agent.

- Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s.
- Each of these actions is **applicable** in s.

TRANSITION MODEL: A description of what each action does is known as the **transition model**

- A function RESULT(s,a) that returns the state that results from doing action a in state s.
- The term **successor** to refer to any state reachable from a given state by a single action
- The **state space** of the problem is the set of all states reachable from the *initial state* by any sequence of actions.
- The *state space* forms a **graph** in which the nodes are states and the links between nodes are actions.
- A **path** in the state space is a sequence of states connected by a sequence of actions.

Problem-Solving Agent: Well-defined problems

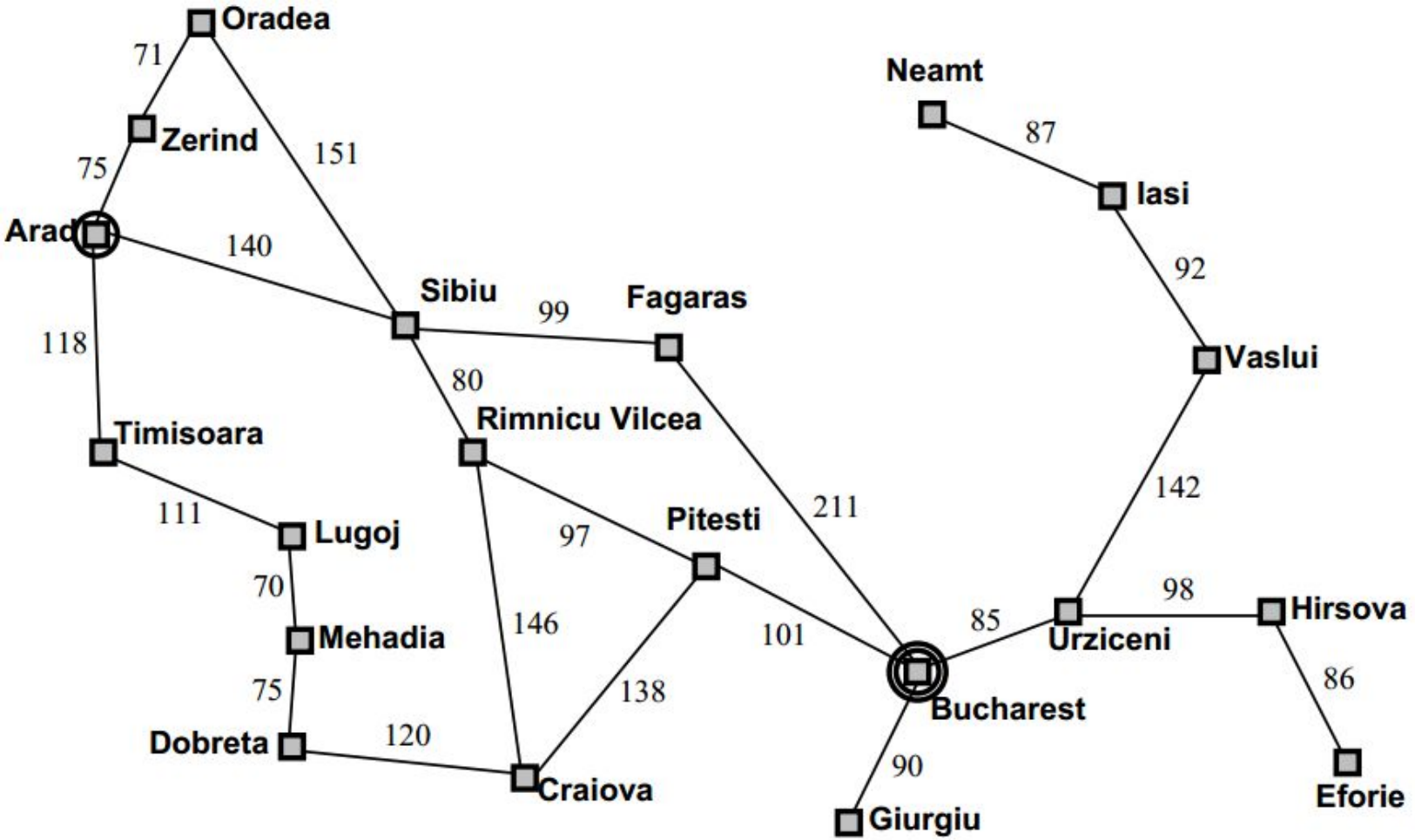
GOAL TEST: The **goal test** determines whether a given state is a goal state.

PATH COST: A **path cost** function that assigns a numeric cost to each path.

- The problem-solving agent chooses a cost function that reflects its own performance measure.
- The **cost of a path** can be described as the sum of the costs of the individual actions along the pa
- The **step cost** of taking action a in state s to reach state s′ is denoted by c(s, a, s′).

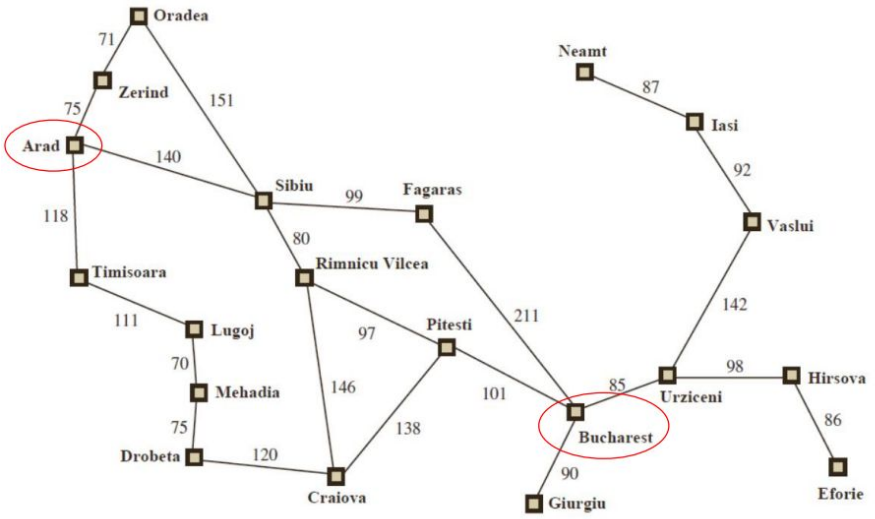
- A **SOLUTION** to a problem is an action sequence that leads from the ***initial state*** to a ***goal state***
- Solution quality is measured by the path cost function, and an **OPTIMAL SOLUTION** has the lowest path cost among all solutions.

Example – Romania Route Map



Problem Example: Travelling in Romania

Agent:
On holiday in Romania, currently in Arad.
Flight leaves tomorrow from Bucharest

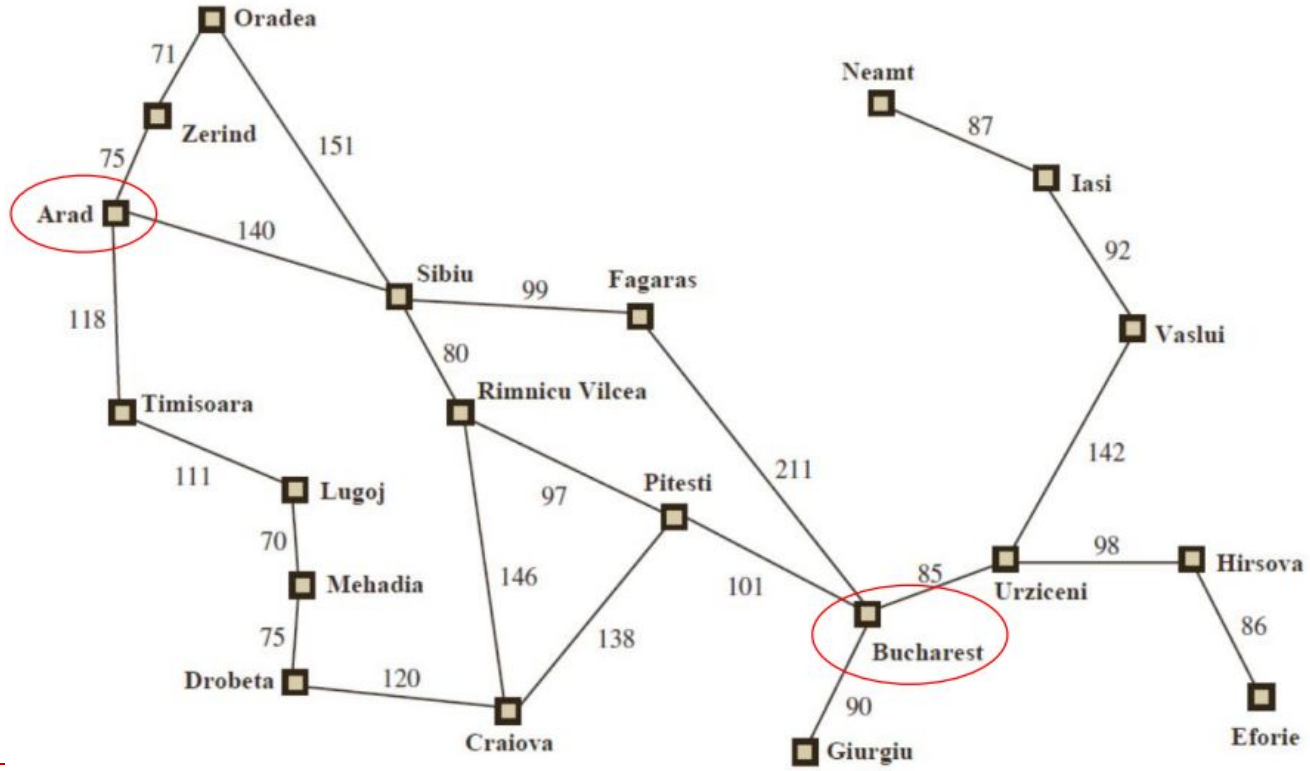


Formulate Goal: Be in Bucharest
Formulate Problem: *States:* various cities ; *Actions:* Drive between cities
Find solution: Sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest

Problem Example: Travelling in Romania

A problem defined by five components:

1. Initial state: In(Arad)



Problem Example: Travelling in Romania

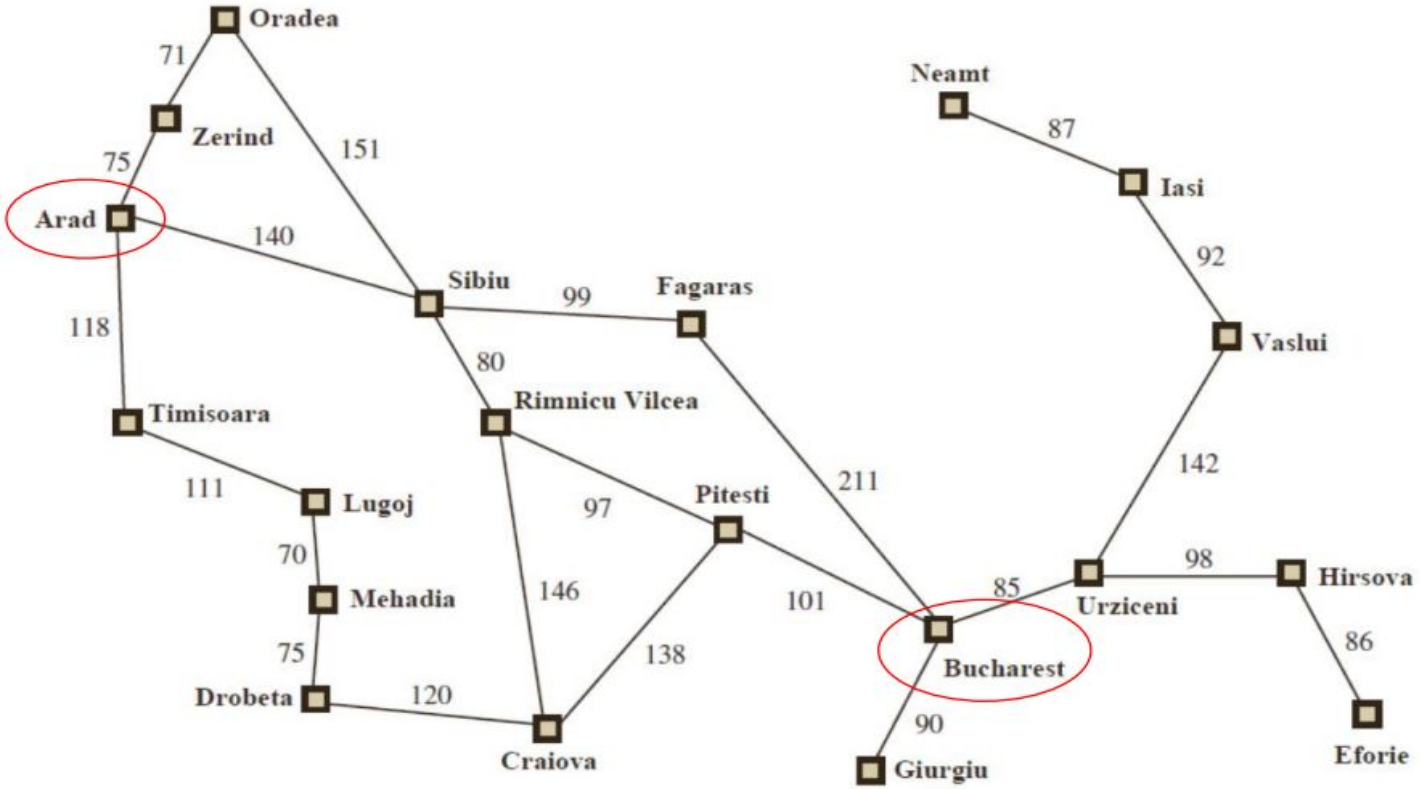
A problem defined by five components:

1. Initial state: In(Arad)

2. Actions: Drive between cities

Example:

In(Arad): Applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.



Problem Example: Travelling in Romania

A problem defined by five components:

1. Initial state: In(Arad)

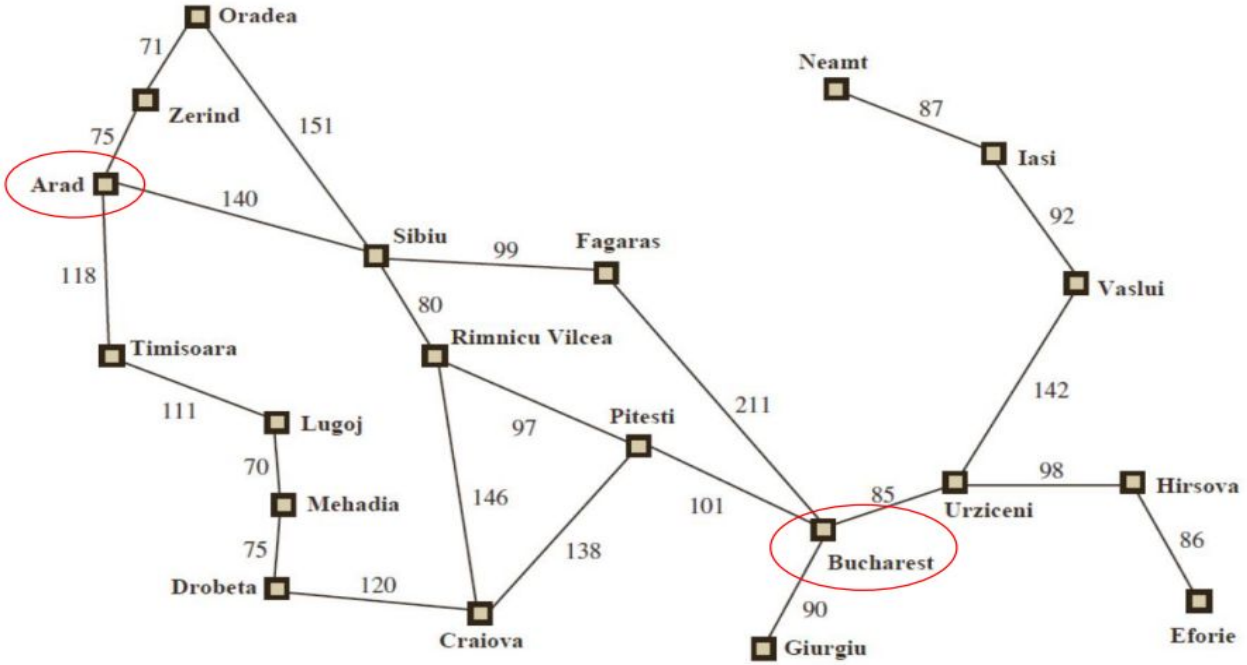
2. Actions: Drive between cities

Example: In(Arad): Applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.

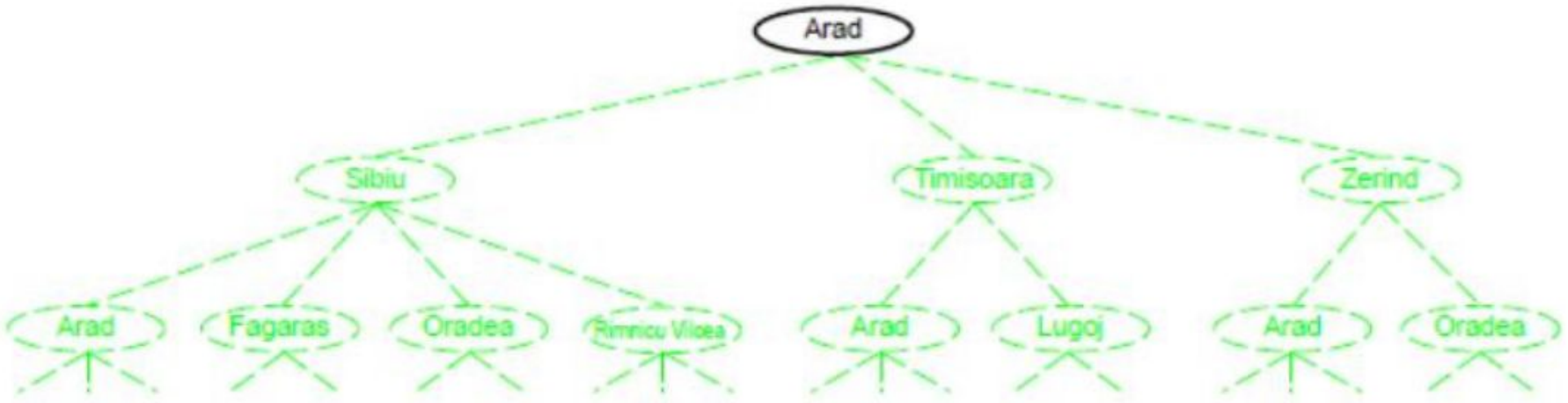
3. Transition model:

Specified by a function RESULT(s, a) that returns the state that results from doing action a in state s

RESULT(In(Arad),Go(Zerind)) = In(Zerind)

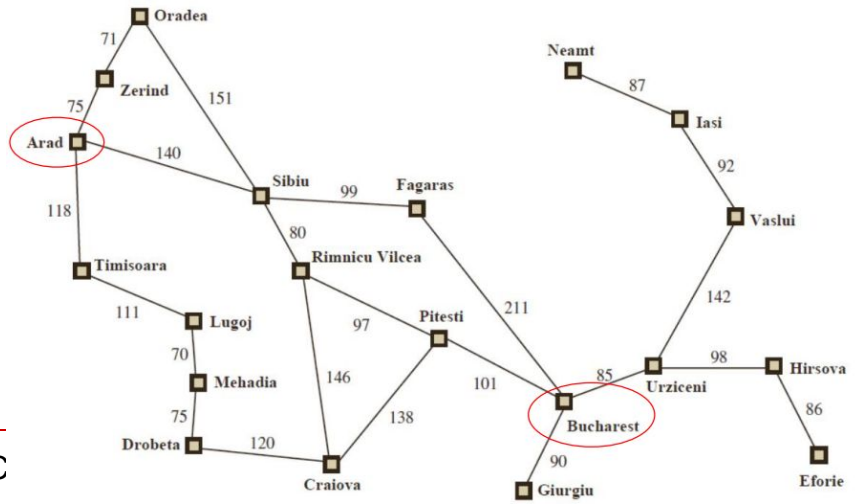


Transition Model:
Partial Search Trees for Travelling in Romania initial state

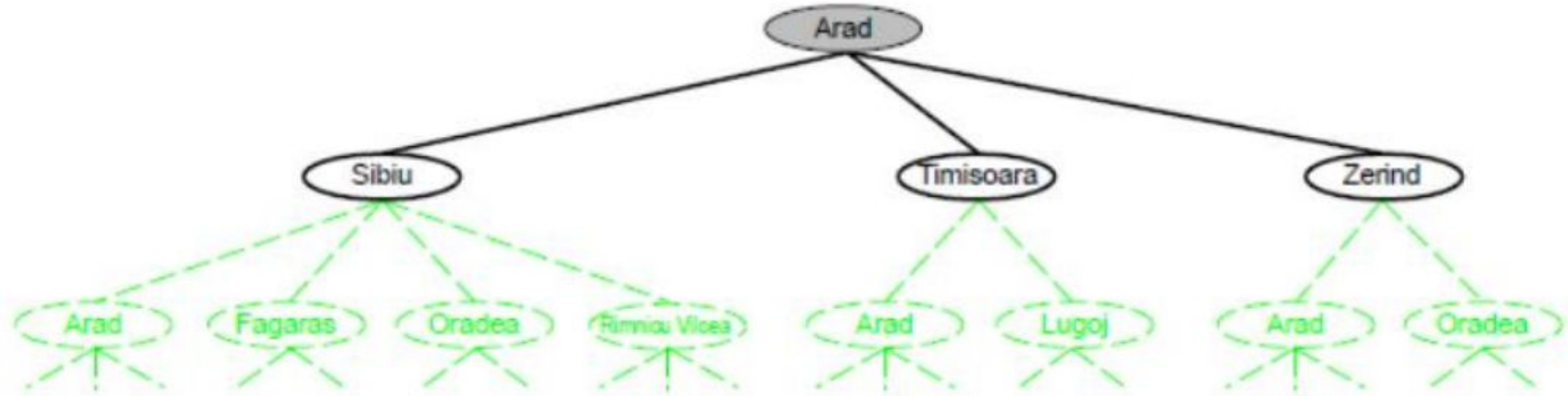


Initial state

Route Map

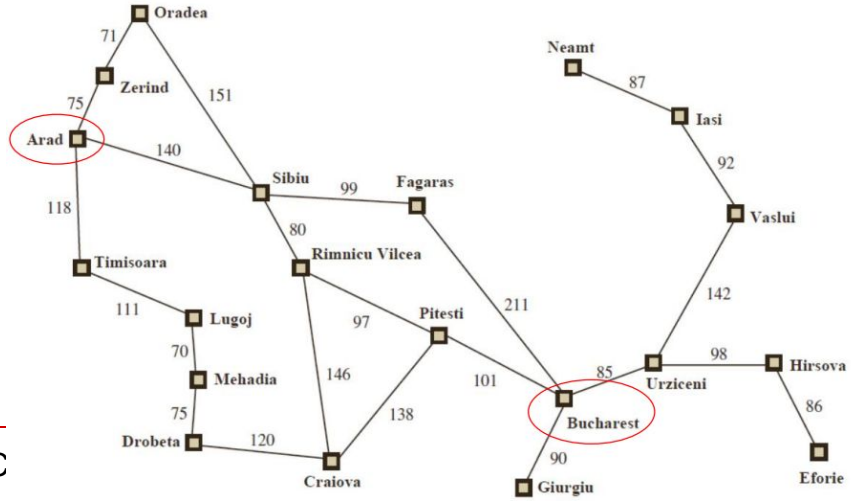


Transition Model:
Partial Search Trees for Travelling in Romania initial state

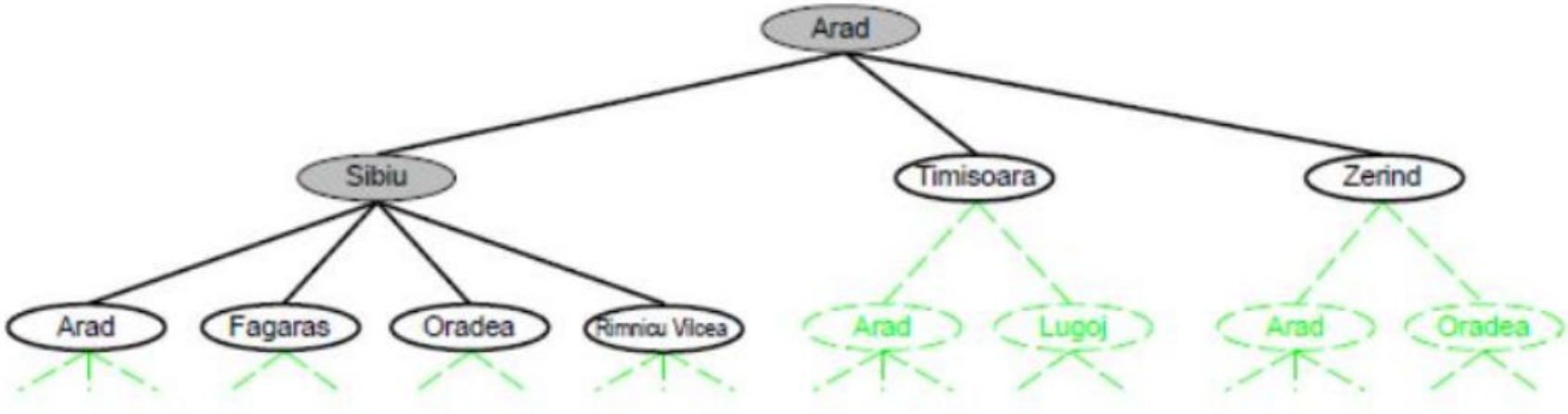


After expanding Arad

Route Map

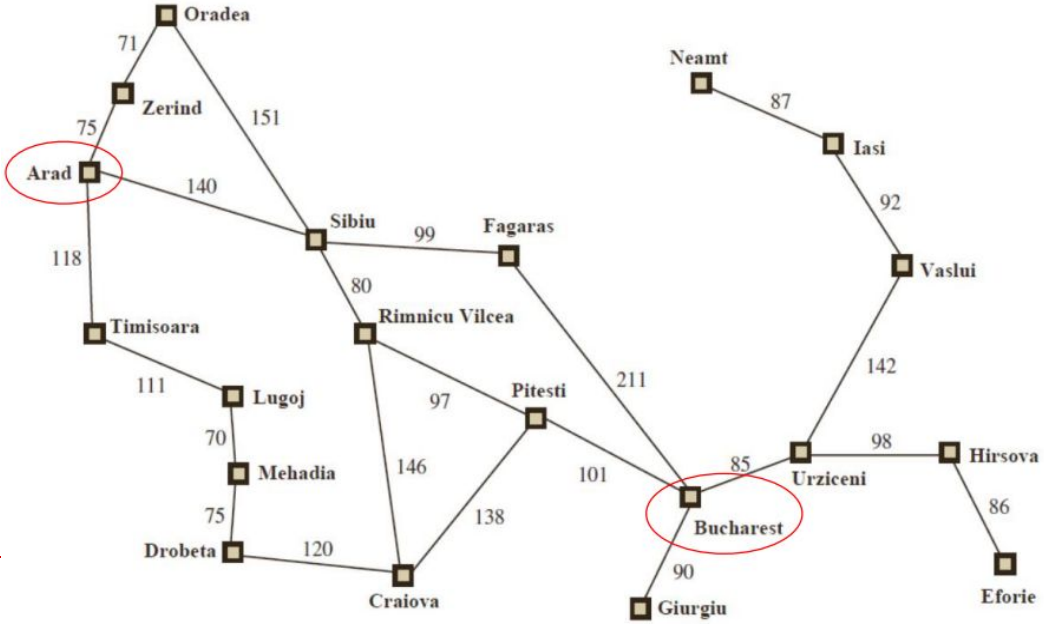


Transition Model:
Partial Search Trees for Travelling in Romania initial state



After expanding Sibiu

Route Map



Problem Example: Travelling in Romania

A problem defined by five components:

1. Initial state: In(Arad)

2. Actions: Drive between cities

Example: In(Arad): Applicable actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.

3. Transition model: Specified by a function RESULT(s, a) that returns the state that results from doing action a in state s

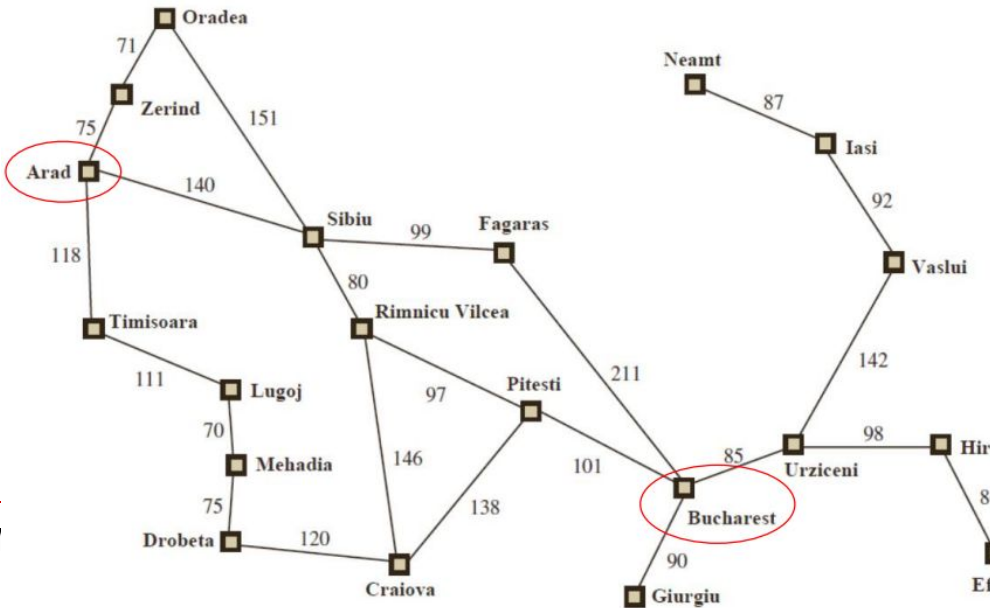
RESULT(In(Arad),Go(Zerind)) = In(Zerind)

4. Goal test: In(Bucharest)

5. Path cost :

Step cost of taking action a in state s to reach state s′ is denoted by c(s, a, s′).

C(In(Arad), Go(Sibiu), In(Sibiu)=140



Problem Solving by Search

- An important aspect of intelligence is goal-based problem solving.
- The solution of many problems can be described by finding a sequence of actions that lead to a desirable goal.

What is Search?

- **Search** is the systematic examination of **states** to find path from the **start/root state** to the **goal state.**
- The set of possible states, together with *operators* defining their connectivity constitute the *search space*.
- The output of a search algorithm is a solution, that is, a path from the initial state to a state that satisfies the goal test.

Problem Solving by Search

A well-defined problem can be described by:

- Initial state
- Operator or successor function - For any state x returns s(x), the set of states reachable from x with one action
- State space - All states reachable from initial state by any sequence of actions
- Path - Sequence through state space
- Path cost - Function that assigns a cost to a path. Cost of a path is the sum of costs of individual actions along the path
- Goal test - Test to determine if at goal state

A simple problem-solving agent

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
              state, some description of the current world state
              goal, a goal, initially null
              problem, a problem formulation

  state ← UPDATE-STATE(state, percept)
  if seq is empty then
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
      if seq = failure then return a null action
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

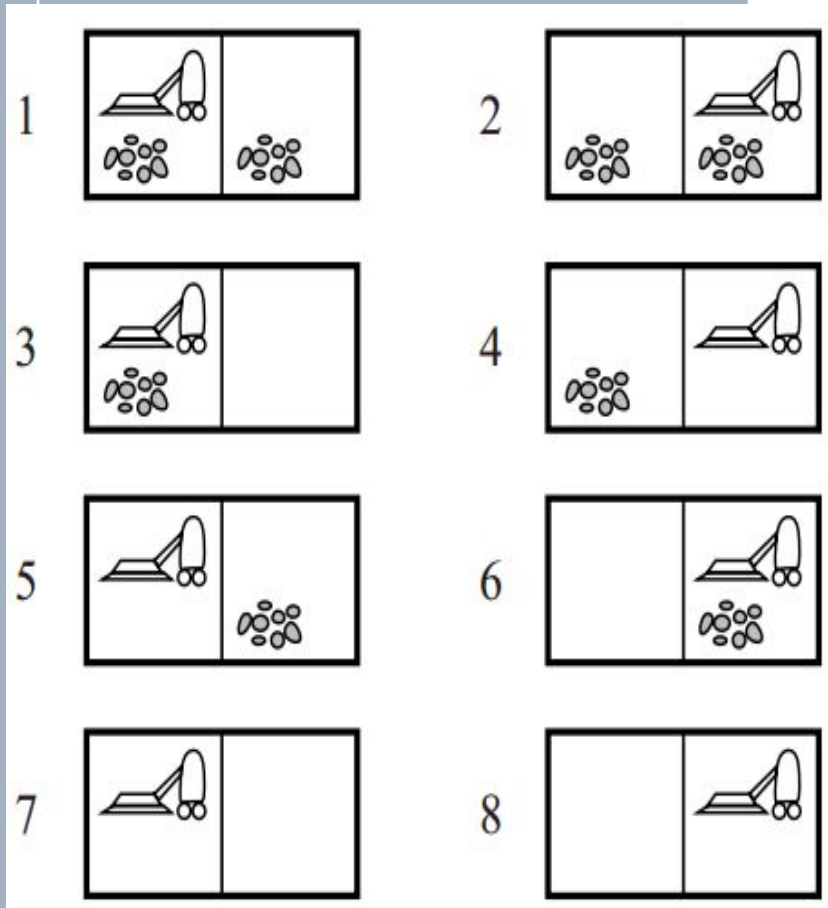
Problem Formulation for the Vacuum Cleaner World

World state space:
2 positions, dirt or no dirt
8 world states

Actions:
Left (L), Right (R), or "PickDust (S)"

Goal:
No dirt in the rooms

Path costs:
One unit per action



Problem Example:
Toy Problem: Vacuum world

States:

- The state is determined by both the agent location and the dirt locations.
- The agent is in one of two locations, each of which might or might not contain dirt.
- Thus, there are $2 \times 2^2 = 8$ possible world states.

Initial state: Any state can be designated as the initial state.

Actions: Each state has just three actions: Left, Right, and "Pick Dust".

Transition model:

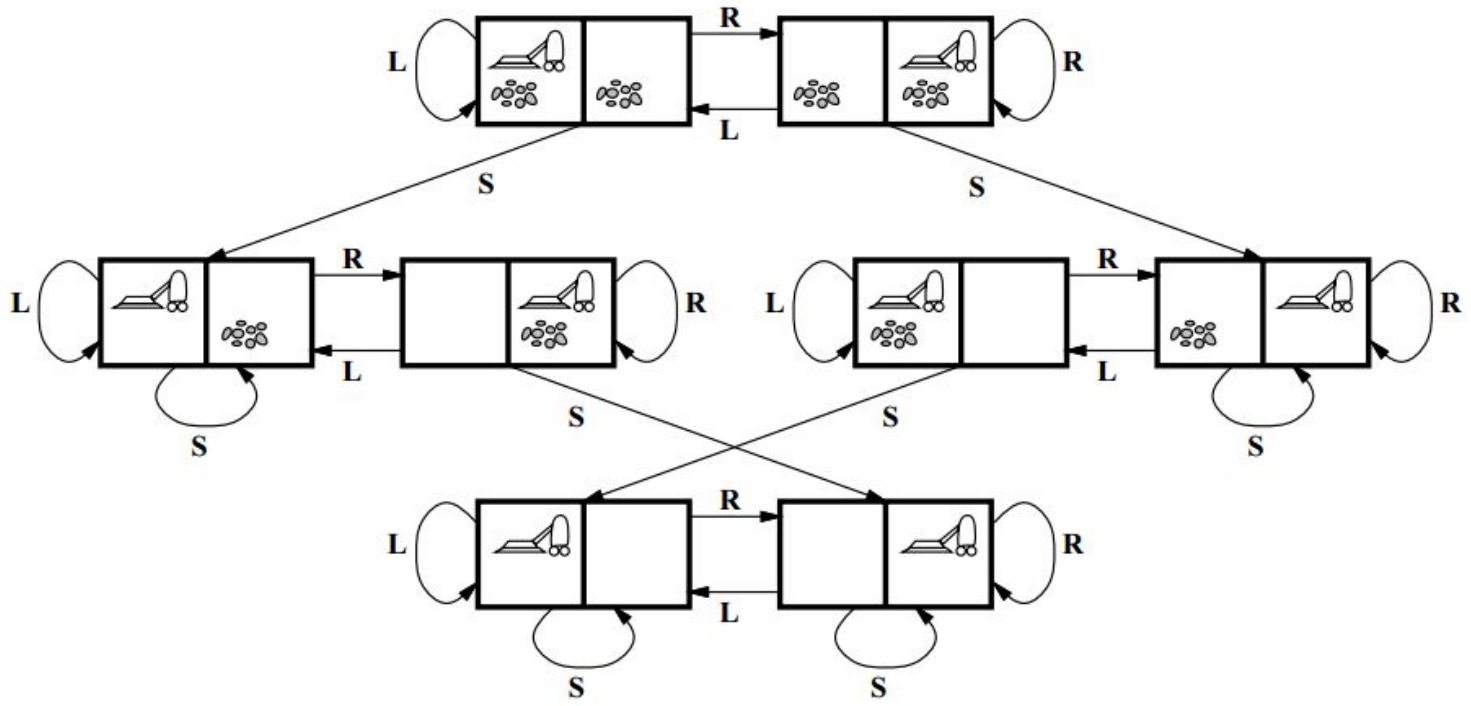
- The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and "Pick Dust" in a clean square have no effect.
- The transition model defines a state space.

Goal test: This checks whether all the squares are clean.

Path cost: Each step costs 1, so the path cost is the number of steps in the path.

The Vacuum Cleaner Problem – State Space

- If the environment is completely observable, the vacuum cleaner always knows where it is and where the dirt is. The solution then is reduced to searching for a path from the initial state to the goal state.



Problem: Example 8 Puzzle

5	4	
6	1	8
7	3	2

Initial State

	1	2
3	4	5
6	7	8

Goal State

- The 8-puzzle consists of a 3×3 board with eight numbered tiles and a blank space.
- A tile adjacent to the blank space can slide into the space.
- The object is to reach a specified goal state.

Problem: Example 8 Puzzle

5	4	
6	1	8
7	3	2

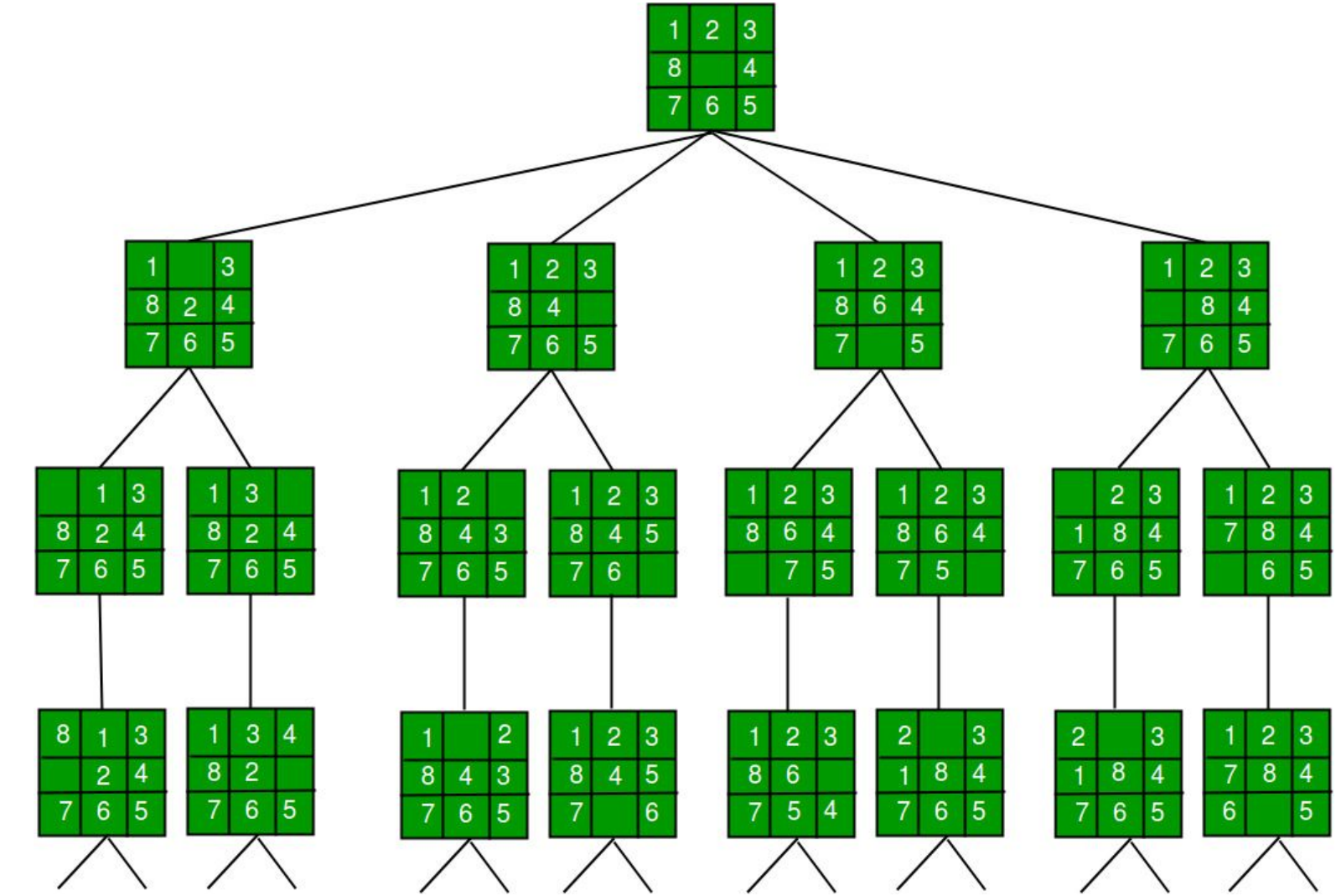
Initial State

	1	2
3	4	5
6	7	8

Goal State

States: A state specifies the location of each of the eight tiles and the blank in one of the nine squares.
Initial state: Any state can be designated as the initial state.
Actions: Movements of the blank space Left, Right, Up, or Down.
Transition model: Given a state and action, this returns the resulting state.
Goal test: The current state matches a Goal state
Path Cost: Each move of the blank costs 1

8 Puzzle – Partial State Space



Question

You are given two jugs, a 4-litre one and a 3-litre one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 liters of water into 4-litre jug. Give the initial state, Goal state, operators and path cost function for the problem and also write the state space diagram.

Water Jug Problem

You are given two jugs, a 4-litre one and a 3-litre one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 liters of water into 4-litre jug. Give the initial state, Goal state, operators and path cost function for the problem and also write the state space diagram.

Input: X = 4, Y = 3, Z = 2

Output: {(0, 0), (0, 3), (3, 0), (3, 3), (4, 2), (0, 2)}

Explanation:

- Fill the 4-litre jug completely with water.
- Empty water from 4-litre jug into 3-litre (leaving 1L water in 4L jug and 3L completely full).
- Empty water from 3L.
- Pour water from 4L jug into 3L jug (4L being completely empty and 1L water in 3L litre jug)
- Fill the 4L jug with water completely again.
- Transfer water from 4L jug to 3L jug, resulting in 2L water in 4L jug.

Problem Statement

Given two water jugs with capacities **X** and **Y** litres. Initially, both the jugs are empty. Also given that there is an infinite amount of water available. The jugs do not have markings to measure smaller quantities.
One can perform the following operations on the jug:

- Fill any of the jugs completely with water.
- Pour water from one jug to the other until one of the jugs is either empty or full,
  (X, Y) -> (X – d, Y + d)
- Empty any of the jugs
- The task is to determine whether it is possible to measure **Z** litres of water using both jugs. And if true, print any of the possible ways.

Example 2:
Input: X = 3, Y = 5, Z = 4
~~**Output:** 6~~
Explanation:

- Fill a 5-litres jug to its maximum capacity.
- Transfer 3 litres from 5-litre jug to 3-litre jug.
- Empty the 3-litre jug.
- Transfer 2-litres from 5-litre jug to 3-litres jug.
- Fill a 5-litres jug to its maximum capacity.
- Pour water into a 3L jug from a 5L jug until it’s full.

Water Jug Problem

- State: (x, y) x = 0, 1, 2, 3, or 4 y = 0, 1, 2, 3
- Start state: (0, 0).
- Goal state: (2, n) for any n. such that n<=3
- Attempting to end up in a goal state

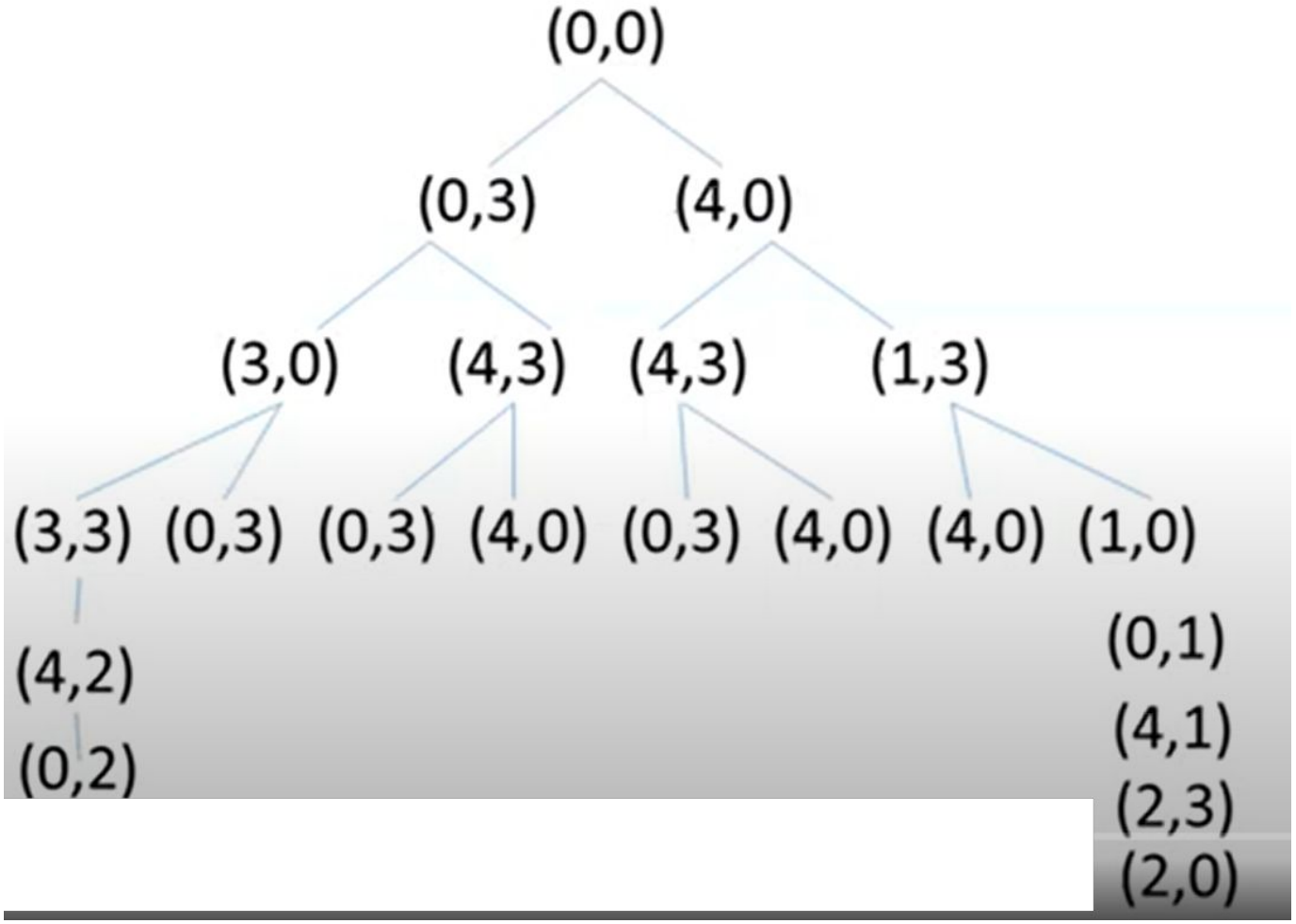
Operators

Sr.	Current State	Next State	Descriptions
1	(x, y) if x < 4	(4, y)	Fill the 4 gallon jug
2	(x, y) if y < 3	(x, 3)	Fill the 3 gallon jug
3	(x, y) if x > 0	(x - d, y)	Pour some water out of the 4 gallon jug
4	(x, y) if y > 0	(x, y - d)	Pour some water out of the 3 gallon jug
5	(x, y) if y > 0	(0, y)	Empty the 4 gallon jug
6	(x, y) if y > 0	(x, 0)	Empty the 3 gallon jug on the ground
7	(x, y) if x + y >= 4 and y > 0	(4, y - (4 - x))	Pour water from the 3 gallon jug into the 4 gallon jug until the 4 gallon jug is full
8	(x, y) if x + y >= 3 and x > 0	(x - (3 - y), 3)	Pour water from the 4 gallon jug into the 3-gallon jug until the 3 gallon jug is full
9	(x, y) if x + y <= 4 and y > 0	(x + y, 0)	Pour all the water from the 3 gallon jug into the 4 gallon jug
10	(x, y) if x + y <=3 and x > 0	(0, x + y)	Pour all the water from the 4 gallon jug into the 3 gallon jug
11	(0, 2)	(2, 0)	Pour the 2 gallons from 3 gallon jug into the 4 gallon jug
12	(2, y)	(0, y)	Empty the 2 gallons in the 4 gallon jug on the ground

One of the possible solution

Gallons in the 4-gallon jug	Gallons in the 3-gallon jug	Rule applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5 or 12
0	2	9 or 11
2	0	--

Water Jug Problem



Question

Give the complete problem formulation for the following.

Choose a formulation that is precise enough to be implemented.

i) You have 3 jugs, measuring 12 gallons, 8 gallons, and 3 gallons, and a water faucet/tap. You can fill the jugs up or empty them out from one to another or onto the ground. You need to measure out exactly one gallon.

Answer

State Space: all configurations of the three jugs with different amounts of water in each jug.

Initial State: 1 empty 12-gallon jug, 1 empty 8 gallon jug, 1 empty 3 gallon jug

Actions:

☐ Action 1: Fill a jug from the faucet until the jug is full

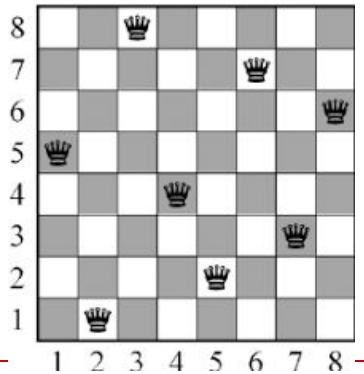
☐ Action 2: Dump water from one jug A into another jug B until B is full

☐ Action 3: Dump all the water from one jug onto the ground Goal: A jug has 1 gallon of water in it and the other jugs are empty Cost: A reasonable approach would be to make the cost of each action a unit cost so that all actions have the same cost. Another possibility is to define the cost of an action to be equal to the number of gallons of water transferred by the action. Furthermore, one could think of making the cost of Action 3 (dumping water onto the group) higher to try to minimize wasting water.

8-queens problem

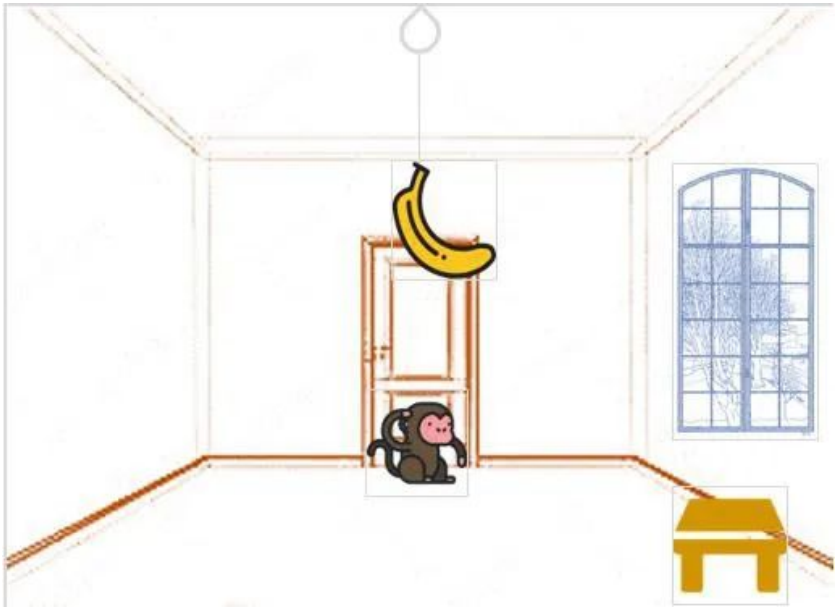
The goal of the 8-queens problem is to place eight queens on a chessboard such that no queen attacks any other. (A queen attacks any piece in the same row, column or diagonal.)

- States: Any arrangement of 0 to 8 queens on the board is a state.
- Initial state: No queens on the board.
- Actions: Add a queen to any empty square.
- Transition model: Returns the board with a queen added to the specified square.
- Goal test: 8 queens are on the board, none attacked.



Question

A 3-foot-tall monkey is in a room where some bananas are suspended from the 8-foot ceiling. She would like to get the bananas and specifically end up on the ground with the bananas. The room contains two stackable, movable, climbable 3-foot-high crates, which you can call A and B. The monkey is initially on the ground, as are both of the crates, and nothing is under the bananas initially. Assume that the monkey wants to accomplish their task with the fewest possible actions. Give a complete problem formulation as a search problem, precise enough to be implemented.



Answer

- **State:**
  - Monkey's current position (initially at the floor level, 0 feet).
  - Banana's position (hanging 8 feet above the floor).
  - Positions of the two crates (initially at the floor level, each at 0 feet).
  - Ceiling height (8 feet).
  - Monkey's current height above the floor (0 feet).
- **Goal:**
  - The monkey has successfully positioned the crates in such a way that it can safely climb onto them to reach the bananas hanging from the 8-foot ceiling.
  - The monkey has retrieved at least one banana from the bunch.
- **Actions:**
  - Move crate A: Slide crate A horizontally to any desired position within the room.
  - Move crate B: Slide crate B horizontally to any desired position within the room.
  - Stack crates: Place one crate on top of the other to increase height.
  - Climb crate(s): Climb on top of the crates to reach higher positions.
  - Reach for bananas: Stretch upwards to try to reach the bananas.

Cost function: the number of actions completed

Question

The missionaries and cannibals problem is usually stated as follows. Three missionaries and three cannibals are on one side of a river, along with a boat that can hold one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint (Amarel, 1968).

a. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space.

Question

The missionaries and cannibals problem is as follows. Three missionaries and three cannibals are on one side of a river, along with a boat. The boat can hold one or two people (and obviously cannot be paddled to the other side of the river with zero people in it). The goal is to get everyone to the other side, without ever leaving a group of missionaries outnumbered by cannibals. Your task is to formulate this as a search problem.

(a) Define a state representation for the given problem.

(b) Give the initial and goal states in this representation.

(c) Define the successor function in this representation.

(d) What is the cost function in your successor function?

(e) What is the total number of reachable states?

Note: If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Misssionaries

Missionaries and cannibals problem

- Three missionaries and three cannibals wish to cross the river.
- They have a small boat that will carry up to two people.
- Everyone can navigate the boat.
- If at any time the Cannibals outnumber the Missionaries on either bank of the river, they will eat the Missionaries.
- **How can everyone get across the river without the missionaries risking being eaten?**

Missionaries and cannibals problem

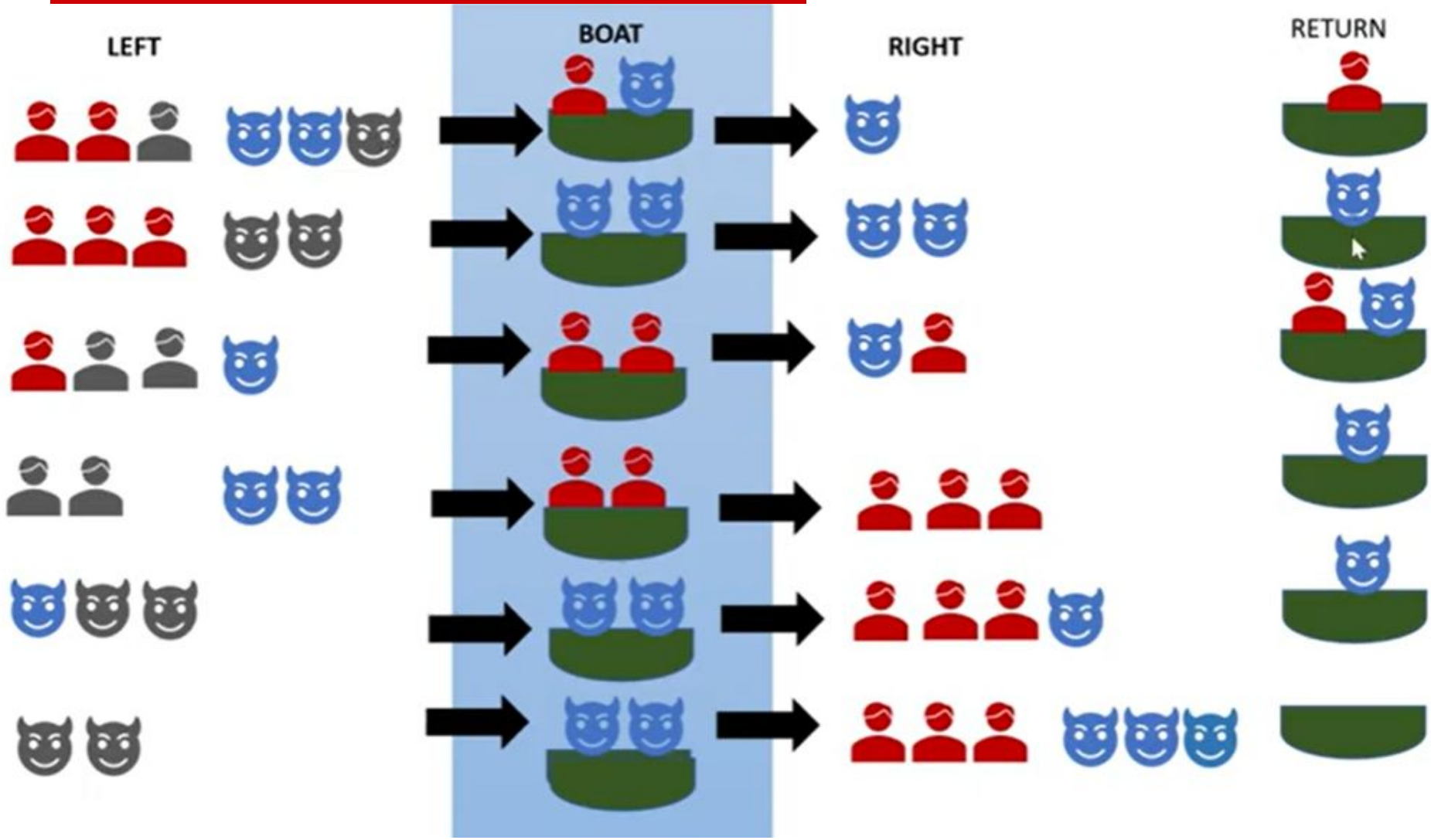
- Initial State: 3 missionaries, 3 cannibals and the boat are on Left side.
  L <MMMCCC>
- Goal: Move all the missionaries and cannibals across the river.
  R<MMMCCC>
- Constraint: Missionaries can never be outnumbered by cannibals on either side of river, or else the missionaries are killed.
- Operators: Move boat from left to right
  Move boat from right to left

Missionaries and cannibals problem

Production Rules

Rule	State	Description
1	(0,M)	One missionary sailing the boat from Left to Right
2	(M,0)	One missionary sailing the boat from Right to Left
3	(M,M) (M,M)	Two missionaries sailing from Left to Right Two missionaries sailing from Right to Left
4	(C,C) (C,C)	Two cannibals sailing from Left to Right Two cannibals sailing from Right to Left
5	(0,C)	One cannibal sailing the boat from Left to Right
6	(C,0)	One cannibal sailing the boat from Right to Left
7	(M,C)	One missionary & one cannibal sailing the boat from Left to Right
8	(C,M)	One missionary & one cannibal sailing the boat from Right to Left

Missionaries and cannibals problem

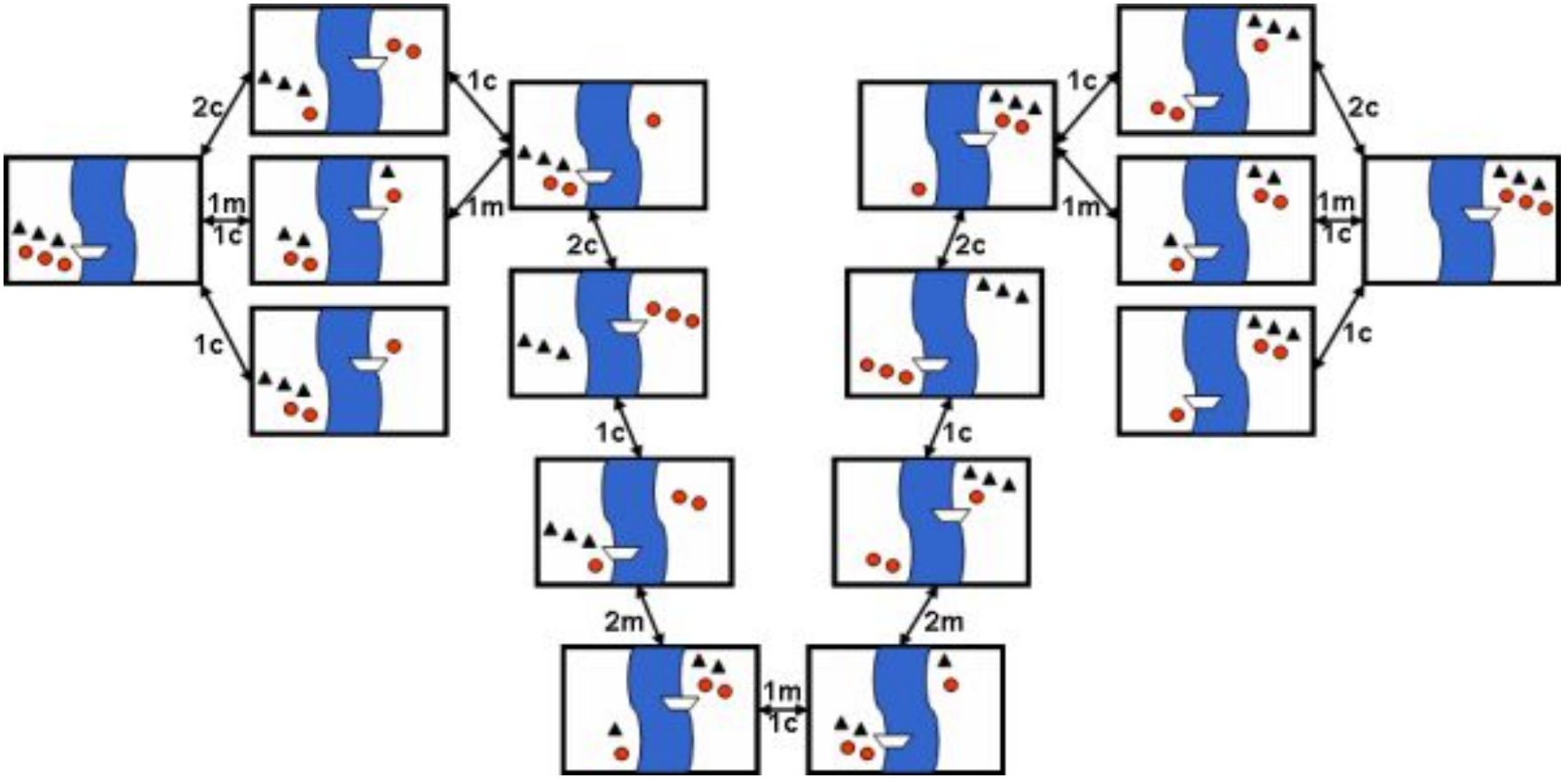


Missionaries and cannibals problem

Solution

ACTION	LEFT	BOAT	RIGHT
A missionary & cannibal	MMMCCC	MC	C
Two cannibals cross over	MMMCC	CC	CC
Two missionaries cross over	MMMC	MM	CM
Two missionaries cross over	MMCC	MM	MMM
Two cannibals cross over	CCC	CC	MMMC
Two cannibals cross over	CC	CC	MMMCCC

Search-space for the Missionaries and Cannibals problem



Question

Your goal is to navigate a robot out of a maze. The robot starts in the center of the maze facing north. You can turn the robot to face north, east, south, or west. You can direct the robot to move forward a certain distance, although it will stop before hitting a wall.

a. Formulate this problem. How large is the state space?

b. In navigating a maze, the only place we need to turn is at the intersection of two or more corridors. Reformulate this problem using this observation. How large is the state space now?

c. From each point in the maze, we can move in any of the four directions until we reach a turning point, and this is the only action we need to do. Reformulate the problem using these actions. Do we need to keep track of the robot's orientation now?

d. In our initial description of the problem we already abstracted from the real world, restricting actions and removing details. List three such simplifications we made

Solution

a. We'll define the coordinate system so that the center of the maze is at $(0, 0)$, and the maze itself is a square from $(-1, -1)$ to $(1, 1)$.

Initial state: robot at coordinate $(0, 0)$, facing North.

Goal test: either $|x| > 1$ or $|y| > 1$ where (x, y) is the current location.

Successor function: move forwards any distance d; change direction robot it facing.

Cost function: total distance moved.

The state space is infinitely large, since the robot's position is continuous.

Solution

b. The state will record the intersection the robot is currently at, along with the direction it's facing. At the end of each corridor leaving the maze we will have an exit node. We'll assume some node corresponds to the center of the maze.

Initial state: at the center of the maze facing North.

Goal test: at an exit node.

Successor function: move to the next intersection in front of us, if there is one; turn to face a new direction.

Cost function: total distance moved.

There are $4n$ states, where n is the number of intersections.

c. Initial state: at the center of the maze.

Goal test: at an exit node.

Successor function: move to next intersection to the North, South, East, or West.

Cost function: total distance moved.

We no longer need to keep track of the robot's orientation since it is irrelevant to predicting the outcome of our actions, and not part of the goal test. The motor system that executes this plan will need to keep track of the robot's current orientation, to know when to rotate the robot.

Solution

d. State abstractions:

(i) Ignoring the height of the robot off the ground, whether it is tilted off the vertical.

(ii) The robot can face in only four directions.

(iii) Other parts of the world ignored: possibility of other robots in the maze, the weather in the Caribbean.

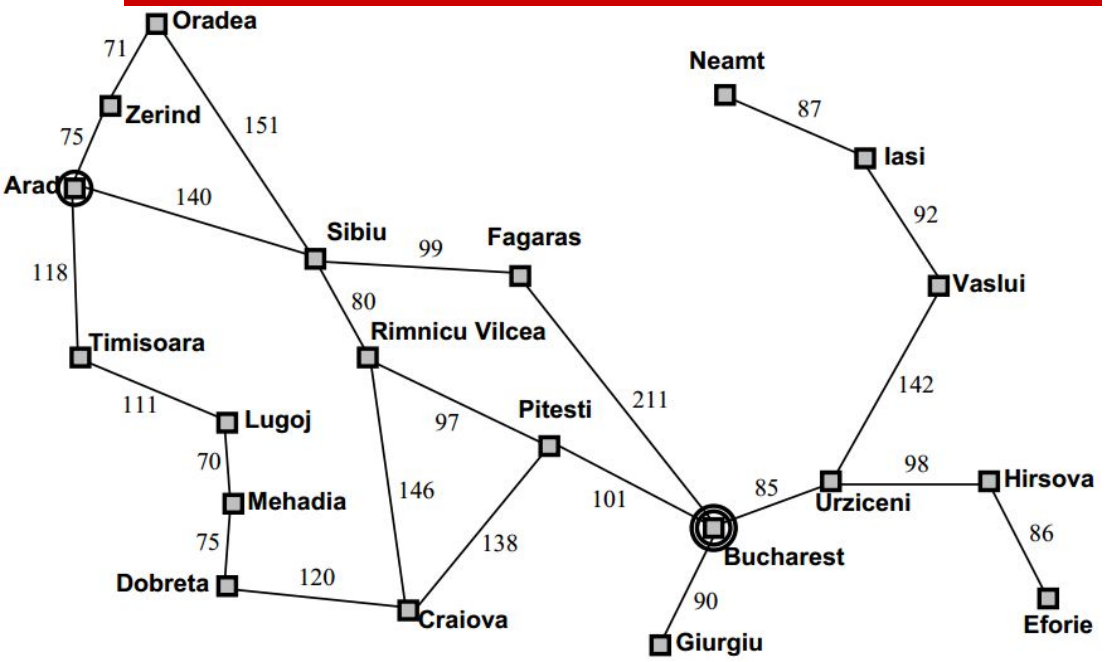
Action abstractions:

(i) We assumed all positions we safely accessible: the robot couldn't get stuck or damaged.

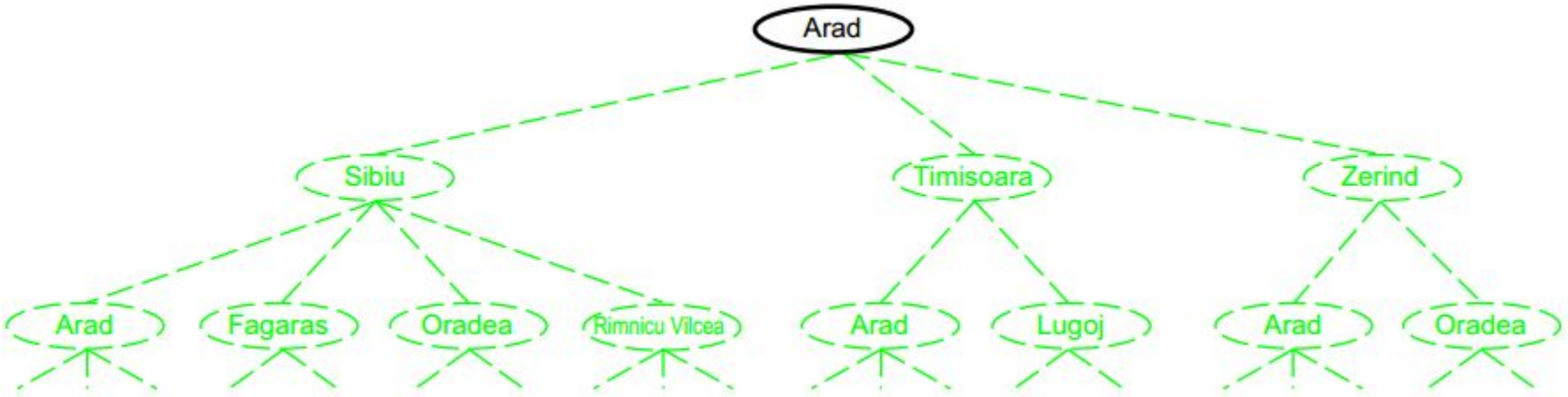
(ii) The robot can move as far as it wants, without having to recharge its batteries.

(iii) Simplified movement system: moving forwards a certain distance, rather than controlled each individual motor and watching the sensors to detect collisions.

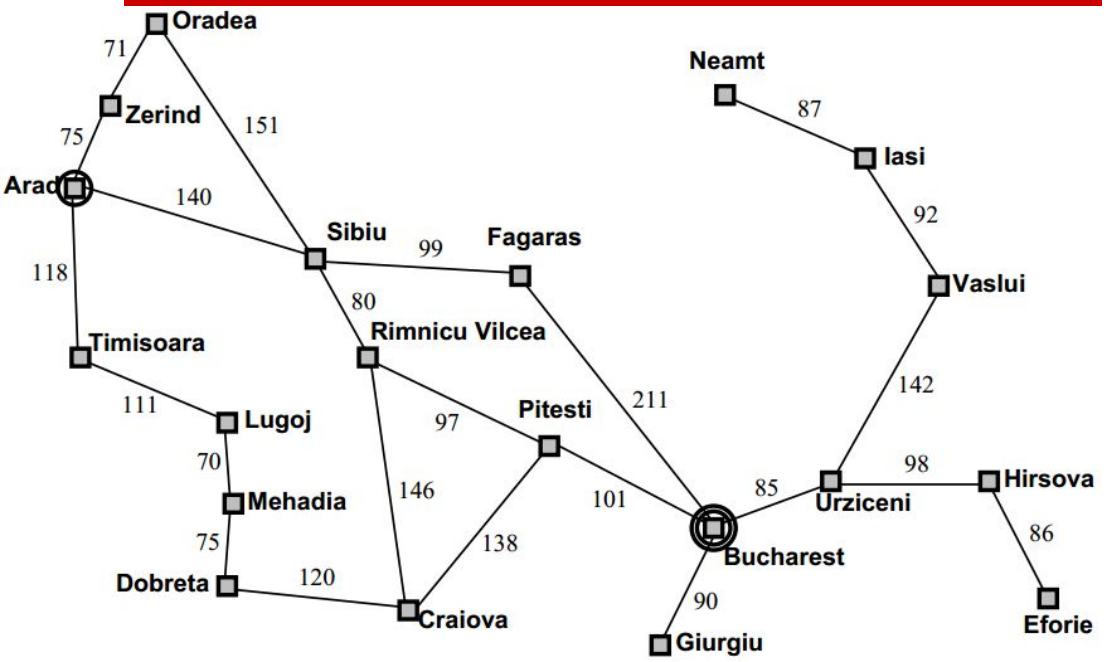
Searching for Solutions - Tree Search



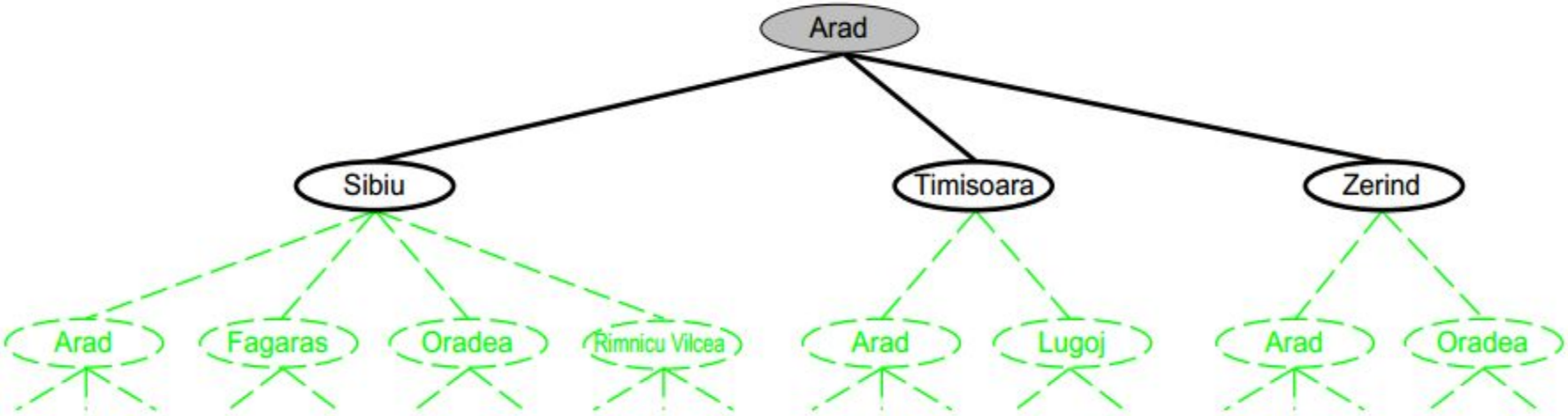
Route Map of Romania City



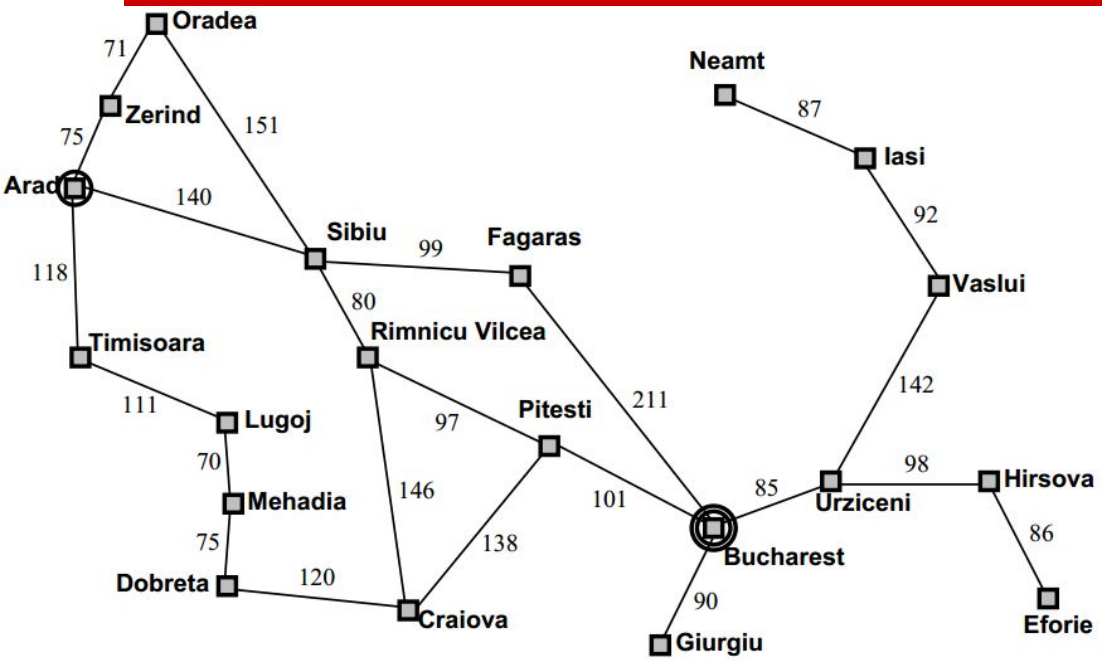
Searching for Solutions - Tree Search



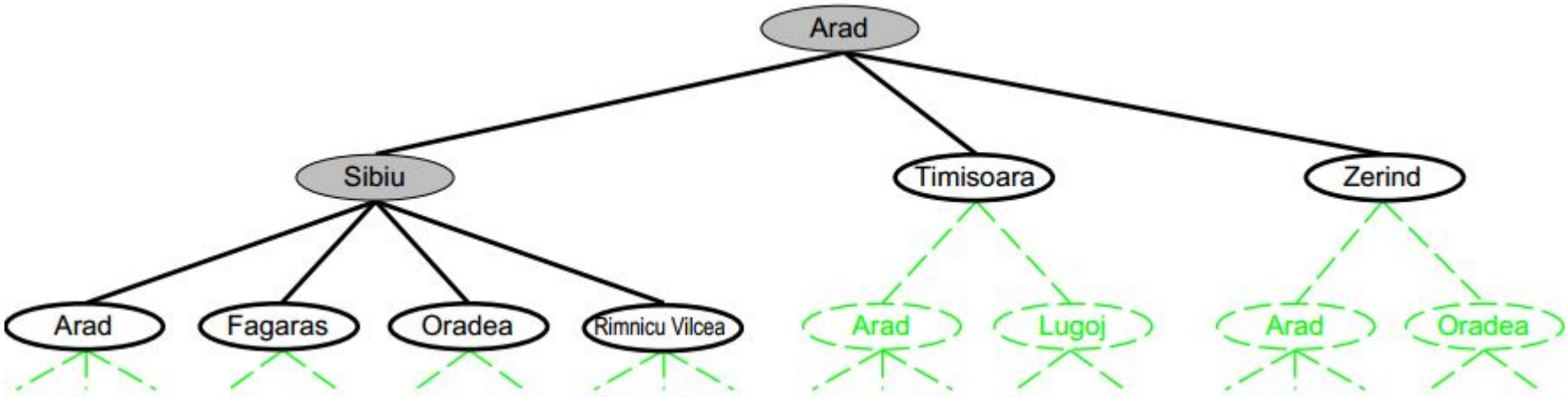
Route Map of Romania City



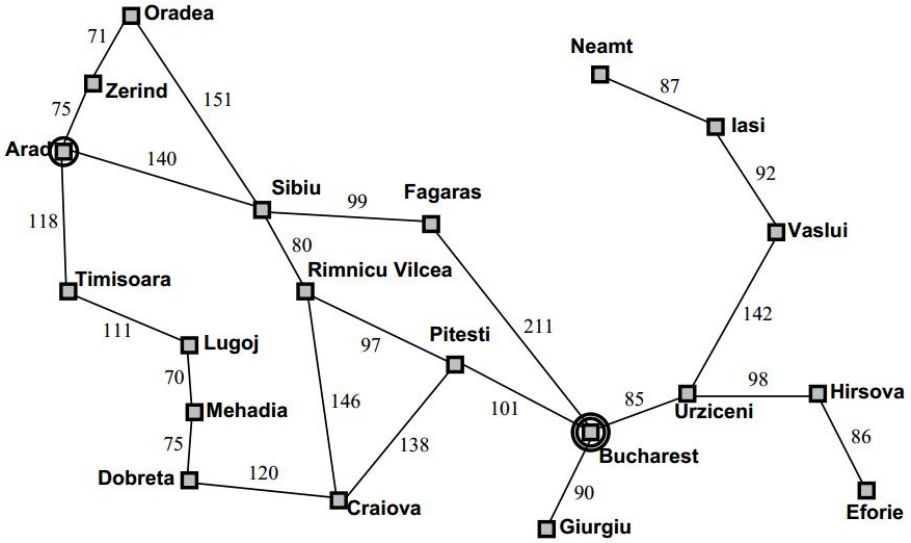
Searching for Solutions - Tree Search



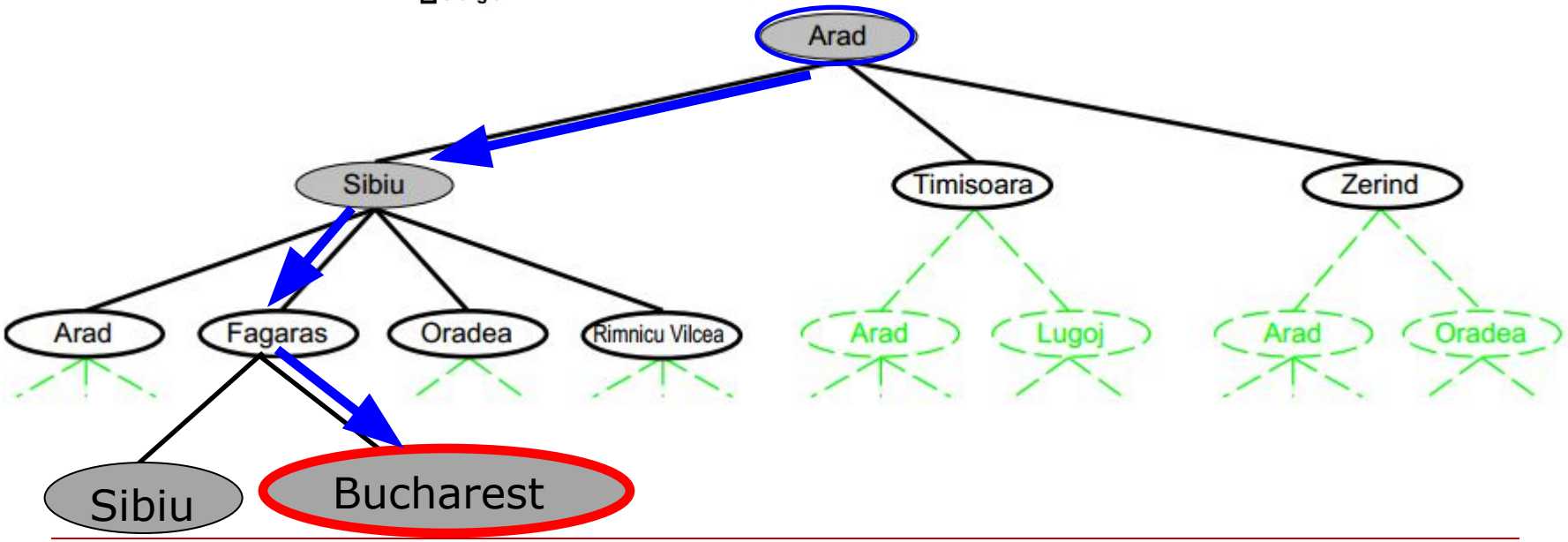
Route Map of Romania City



Searching for Solutions - Tree Search



Route Map of Romania City



Tree Search Algorithms

Basic idea:

- Offline, simulated exploration of state space
- By generating successors of already-explored states (a.k.a. expanding states)

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        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
```

Graph Search Algorithms

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure
initialize the frontier using the initial state of *problem*
initialize the explored set to be empty
loop do
if the frontier is empty **then return** failure
choose a leaf node and remove it from the frontier
if the node contains a goal state **then return** the corresponding solution
add the node to the explored set
expand the chosen node, adding the resulting nodes to the frontier
only if not in the frontier or explored set

Search Strategies

1. Uniformed or Blind Search:
   Uninformed Search Strategies have no additional information about states beyond that provided in the problem definition.

2. Informed or Heuristic Search
   Search Strategies that know whether one non goal state is "more promising" than another are called

Criteria for Search strategies

Strategies are evaluated along the following dimensions:

- Completeness - Is the strategy guaranteed to find a solution when there is one?
- Time complexity - How long does it take to find a solution?
- Space complexity - How much memory does the search require?
- Optimality - Does the strategy find the best solution (with the smallest possible path cost)?

State space features governing complexity:

b: Branching factor.

d: Depth of shallowest goal node.

m: Maximum length of any path in the state space.

Summary

- Agents interact with environments through actuators and sensors
- The agent function describes what the agent does in all circumstances
- The performance measure evaluates the environment sequence
- A perfectly rational agent maximizes expected performance
- Agent programs implement (some) agent functions
- PEAS descriptions define task environments
- Environments are categorized along several dimensions:

Observable? Deterministic? Episodic? Static? Discrete? Single-agent?

- Several basic agent architectures exist:

reflex, reflex with state, goal-based, utility-based

Summary: Intelligent Agents

- An **agent** perceives and acts in an environment, has an architecture, and is implemented by an agent program.
- Task environment – **PEAS (P**erformance**, E**nvironment, **A**ctuators, **S**ensors**)**
- The most challenging environments are inaccessible, nondeterministic, dynamic, and continuous.
- An **ideal agent** always chooses the action which maximizes its expected performance, given its percept sequence so far.
- An **agent program** maps from percept to action and updates internal state.
  - **Reflex agents** respond immediately to percepts.
    - simple reflex agents
    - model-based reflex agents
  - **Goal-based agents** act in order to achieve their goal(s).
  - **Utility-based agents** maximize their own utility function.
- All agents can improve their performance through **learning.**

Summary

- An agent is something that perceives and acts. It consists of an architecture and an agent program.
- An ideal rational agent always takes the action that maximizes its performance given the percept sequence and its environment knowledge.
- There are a variety of agent designs:
  - Reflex agents respond immediately to percepts.
  - Goal-based agents work towards goals.
  - Utility-based agents try to maximize their reward.
  - Learning agents improve their behavior over time.
- Some environments are more demanding than others . .
- . . . your own, and that of James Bond, are the most difficult.

Thanks for Listening