Artificial Intelligence

Module 2: Automated Problem Solving

PART 2.2: Complex Problems and AI

Dr. Chandra Prakash

(Slides adapted from StuartJ. Russell, B Ravindran, Mausam, Prof. Pallab Dasgupta, Prof. Partha Pratim Chakrabarti, Saikishor Jangiti)

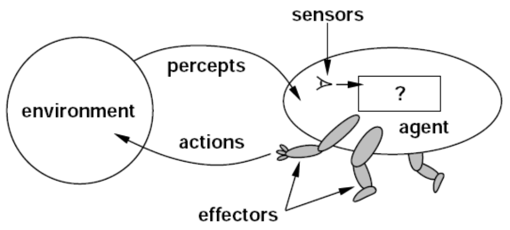
Module 2: Automated Problem Solving

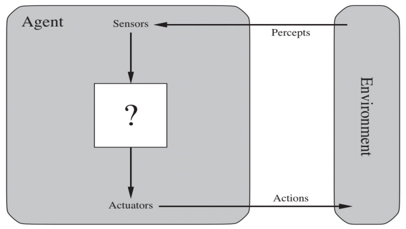
- PART 2.1: Intelligent Agent & Environment
- PART 2.2: Complex Problems and AI
- PART 2.3: Problem Solving Methods

PART 2.2: Complex Problems and AI

- Easy AI Problems : Game Problem
- Approach for Tic Tac Toe
- Famous Problems for Illustrating AI Concepts
  - Tic-Tac-Toe
  - Travelling Salesman Problem
  - Water Jug Problem
  - 8-puzzle problem
  - Missionaries and Cannibals problem
  - Towers of Hanoi
  - Monkey and Bananas problem
  - Cryptarithmetic
  - Sudoku
  - 8 Queens problem
- AI based Automated Problem solving approach

Intelligent Agents Recap





- Rational Agent
- Task Environment: PEAS
- Type of Agents
  - Simple Reflex Agent
  - Model based Agent
  - Goal based Agent
    - Problem Solving Agents
  - Utility based Agent
  - Learning based Agent

AI Problem Areas /Tasks

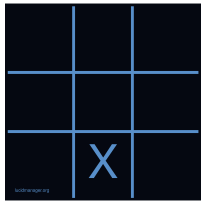
- **1st Generation of AI :**
  Formal cognitive Tasks
  - Game
    - Tic-Tac-Toe
    - Chess
    - Checkers
    - Go
  - Mathematics
    - Logic
    - Geometry
    - Calculus
    - Proving properties of programs
- **2nd Generation :**
  Expert Tasks
  - Engineering
    - Design
    - fault finding
    - Manufacturing planning
  - Medical
    - Diagnosis
    - Medical Image Analysis
  - Financial
    - Stock market predictions
- **3rd Generation of AI :**
  Perceptual Tasks
  - Perception
    - Vision
    - Speech
  - Natural Language
    - Understanding
    - Generation
    - Translation
  - Robot Control

What's involved in Intelligence?

- Ability to interact with the world (speech, vision, motion, manipulation)
- Ability to model the world and to reason about it
- Ability to learn and to adapt
- Goal of AI
  - To build systems that exhibit intelligent behavior
  - To understand intelligence in order to model it
  - Our aim is to solve all type of problems in the world

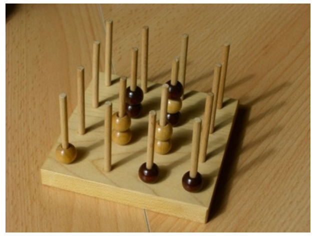
Fomal cognitive Tasks / AI Problems : Game Problem

- Lets PLAY some Games
  - Why Games ?????
- Tic Tac Toe
  - https://g.co/kgs/XCLRWG
- Chess



Tic Tac Toe

- Three programs are presented :
  Series increase
  - Their complexity
  - Use of generalization
  - Clarity of their knowledge
  - Extensability of their approach



Introductory Problem: Tic-Tac-Toe

X		X
	o	

```
# Data structure for supporting uniform cost search.
class PriorityQueue:
    def  __init__(self):
        self.DONE = -100000
        self.heap = []
        self.priorities = {}  # Map from state to priority

    # Insert |state| into the heap with priority |newPriority| if
    # |state| isn't in the heap or |newPriority| is smaller than the existing
    # priority.
    # Return whether the priority queue was updated.
    def update(self, state, newPriority):
        oldPriority = self.priorities.get(state)
        if oldPriority == None or newPriority < oldPriority:
            self.priorities[state] = newPriority
            heapq.heappush(self.heap, (newPriority, state))
            return True
        return False

    # Returns (state with minimum priority, priority)
    # or (None, None) if the priority queue is empty.
    def removeMin(self):
        while len(self.heap) > 0:
            priority, state = heapq.heappop(self.heap)
            if self.priorities[state] == self.DONE: continue  # Outdated priority, skip
            self.priorities[state] = self.DONE
            return (state, priority)
        return (None, None) # Nothing left...

############################################################
# Simple examples of search problems to test your code for Problem 1.

# A simple search problem on the number line:
# Start at 0, want to go to 10, costs 1 to move down, 2 to move up.
class NumberLineSearchProblem:
    def startState(self): return 0
    def isGoal(self, state): return state == 10
    def succAndCost(self, state): return [('West', state-1, 1), ('East', state+1, 2)]

# Function to create search problems from a graph.
# You can use this to test your algorithm.
def createSearchProblemFromString(start, goal, description):
    # Parse the graph
    graph = collections.defaultdict(list)
    for line in description.split("\n"):
        if len(line) == 0 or line.startswith('#'): continue
        # Edge from state a to state b.
        a, b, cost = line.split(" ")
        cost = float(cost)
        # Action is the same as the destination state (b).
        graph[a].append((b, b, cost))
```

Approach 1: Tic-Tac-Toe

X		X
	o	

Data Structures:

- Board: 9 element vector representing the board, with 1-9 for each square.
- An element contains the value
  - 0 : blank,
  - 1 : filled by X, or
  - 2 : filled with a O
- Move-Table:
  - A large vector of 19,683 elements (3^9), each element is 9-element vector.

Algorithm:

1. View the vector as a ternary number. Convert it to a decimal number.
2. Use the computed number as an index into Move-Table and access the vector stored there.
3. Set the new board to that vector.

Approach 1 : Tic-Tac-Toe

Comments:
This program is very efficient in time.

1. A lot of space to store the Move-Table.
2. A lot of work to specify all the entries in the Move-Table.
3. Difficult to extend.

Thus not a good AI Technique

Approach 2 : Tic-Tac-Toe

1	2	3
4	5	6
7	8	9

Approach 2 : Tic-Tac-Toe

Program 2:

Data Structure:
- A nine element vector representing the board. Board[X]
- But instead of using 0,1 and 2 in each element, we store 2 for blank, 3 for X and 5 for O

X		X
	o	

Functions:

- Make 2: returns 5 if the centre square is blank. Else any other blank square
- Posswin(p): Returns 0 if the player p cannot win on his next move;
  otherwise it returns the number of the square that constitutes a winning move.
  If the product is 18 (3x3x2), then X can win.
  If the product is 50 (5x5x2) then O can win.
- Go(n): Makes a move in the square n

Strategy:

Turn = 1 Go(1)
Turn = 2 If Board[5] is blank, Go(5), else Go(1)
Turn = 3 If Board[9] is blank, Go(9), else Go(3)
Turn = 4 If Posswin(X) $\neq$ 0, then Go(Posswin(X))
....

Approach 2 : Tic-Tac-Toe

Comments:

1. Not efficient in time, as it has to check several conditions before making each move.
2. Easier to understand the program's strategy.
3. Any bug in programmer tic-tac-toe playing skill will show up in program's play.
3. Hard to generalize.

Approach 3 : Tic-Tac-Toe

8	3	4
1	5	9
6	7	2

15 – (8 + 5)

Approach 3 :New appraoch

- All row, column and diagonal sum is 15
- Make a list, for each player , of the squares in which he/she has played.
- Consider each pair of square owned by that player
- Computer difference between 15 and sum of two square.
  - If difference is -ve or if greater then 9, then the original two square were not collinear and thus can be ignored.
- If square representing the difference is blank, a move there will produce a win.
- No player cant have more then 4 square at a time, so fewer square to compared to.

Approach 3 : Tic-Tac-Toe

Comments:

1. Checking for a possible win is quicker.
2. Human finds the row-scan approach easier, while computer finds the number-counting approach more efficient.

Approach 3 : Tic-Tac-Toe

Program 3:

Structure with 9-element vector representing the board, a list of board positions that could result from the next move, estimate how likely the board position lead to ultimate win for the player to move.

Algorithm:

To decide to the next move , look ahead at the board position that result from each possible move. Decide which position is best.

1. If it is a win, give it the highest rating.
2. Otherwise, consider all the moves the opponent could make next. Assume the opponent will make the move that is worst for us. Assign the rating of that move to the current node.
3. The best node is then the one with the highest rating.

Approach 3 : Tic-Tac-Toe

Comments:

Algo look ahead at various moves that leads to win. It attempts to maximize the likelihood of winning and opponent will try to minimize that ---minimax procedure.

1. Require much more time to consider all possible moves.
2. Could be extended to handle more complicated games.

Introductory Problem: Question Answering

"Mary went shopping for a new coat. She found a red one she really liked. When she got it home, she discovered that it went perfectly with her favourite dress".

Q1: What did Mary go shopping for?

Q2: What did Mary find that she liked?

Q3: Did Mary buy anything?

Question Answering: Approach 1

Program 1: “Mary went shopping for a new coat. She found a red one she really liked. When she got it home, she discovered that it went perfectly with her favourite dress”.

1. Match predefined templates to questions to generate text patterns.
2. Match text patterns to input texts to get answers.

“What did X Y” “What did Mary go shopping for?”

“Mary go shopping for Z”

Z = a new coat

Question Answering: Approach 2

Program 2:
Convert the input text into a structured internal form that attempts to capture the meaning of the sentences.

“Mary went shopping for a new coat. She found a red one she really liked. When she got it home, she discovered that it went perfectly with her favourite dress”.

Structured representation of sentences:

Event2:		Thing1:	
instance:	Finding	instance:	Coat
tense:	Past	colour:	Red
agent:	Mary		
object:	Thing 1		

Question Answering: Approach 3

Program 3:
Background world knowledge:

$$\begin{tikzcd}
 & \text{C finds M} \arrow[dl] \arrow[dr] & \\
\text{C leaves L} & & \text{C buys M} \arrow[d] \\
 & & \text{C leaves L} \arrow[d] \\
 & & \text{C takes M}
\end{tikzcd}$$

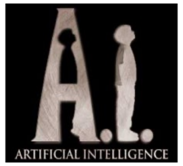
Problem formulation

- Suppose that the agent's sensors give it enough information to tell exactly which state it is in (i.e., the world is accessible);
- Suppose that it knows exactly what each of its actions does.
- Then it can calculate exactly which state it will be in after any sequence of actions.

Problem Solving Agents

- Goal Formulation
  - Organize behavior of the agent
  - Goal
    - set of states in the world where the goal is satisfied
- Problem Formulation
  - What are the actions?
  - What are the states?
- Assumptions about the Task Environment
  - Observable or partially observable?
  - Discrete or Continuous?
  - Deterministic or Stochastic?
  - Static or Dynamic?
  - Episodic or Sequential?
  - Multiple or Single Agent?

- What is the solution to this problem?

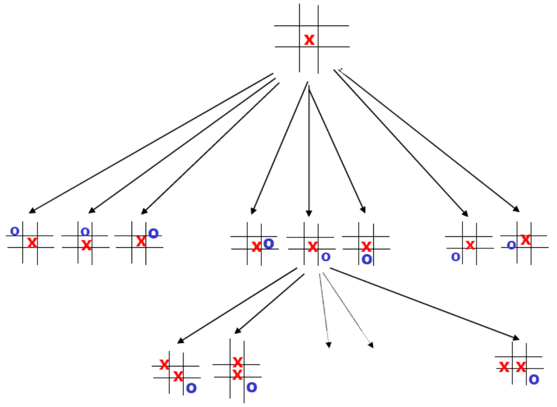


AI Famous Problems

Famous Problems for Illustrating AI Concepts

- Tic-Tac-Toe
- Travelling Salesman Problem
- Water Jug Problem
- 8-puzzle problem
- Towers of Hanoi
- 8 – Queens problem
- Monkey and Bananas problem
- Missionaries and Cannibals problem
- Cryptarithmetic
- Sudoku

1.Tic-Tac-Toe



2.Travelling Salesman Problem

- A salesman has a list of cities, each of which he must visit exactly once.
  - direct roads between each pair of cities on the list.
- Find the route the salesman should follow for the shortest possible round trip that both starts and finishes at any one of the cities.
- The total time required to perform this search is proportional to N!
  - Combinatorial explosion. With 10 cities number of possibilities grows to 3,628,800!

3.Water Jug Problem

- You are given two jugs, a 4-gallon one and 3-gallon one.
  - Neither has any measuring marks on it.
  - There is a pump that can be used to fill the jugs with water.

How can you get exactly 2 gallons of water into the 4-gallon jug?

Goal: 2 Gallons Water in the 4 Gallon Jug

4. 8-puzzle problem

- The 8-puzzle is a square tray in which are placed eight square tiles.
  - Each tile has a number on it.
  - A tile that is adjacent to the blank space can be slid into that space.
  - A game consists of a starting position and a specified goal position.
- The goal is to transform the starting position into the goal position by sliding the tiles around.

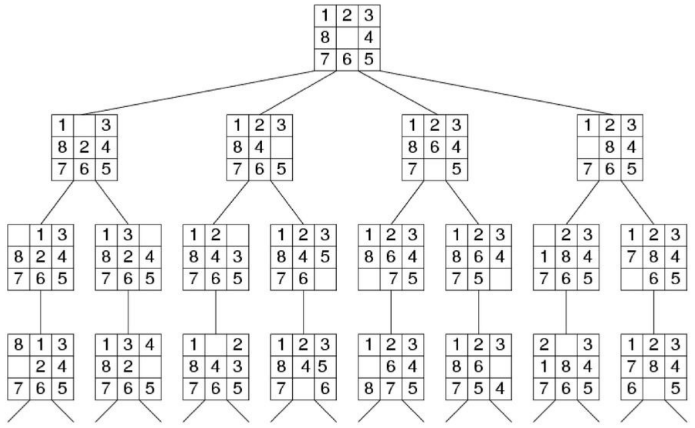
5	4	
6	1	8
7	3	2

Initial State

1	4	7
2	5	8
3	6	

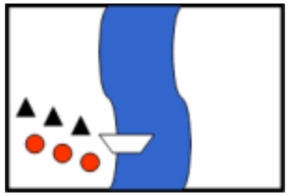
Goal State

4. 8-puzzle problem (Cont..)

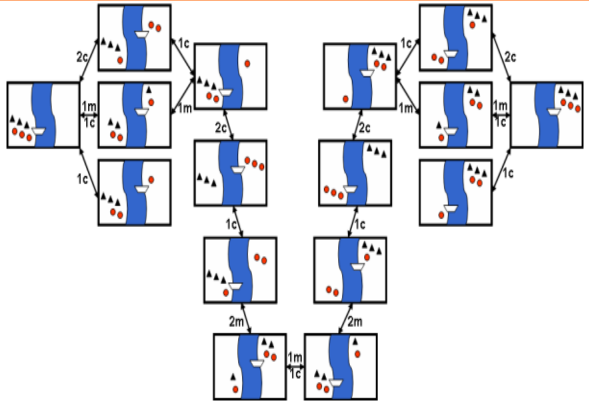


5. Missionaries and Cannibals Problem

- On one bank of a river are three missionaries and three cannibals.
  - There is one boat available that can hold up to two people and that they would like to use to cross the river.
  - If the cannibals ever outnumber the missionaries on either of the river's banks, the missionaries will get eaten.
- How can the boat be used to safely carry all the missionaries and cannibals across the river?

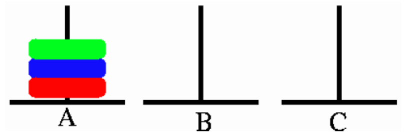


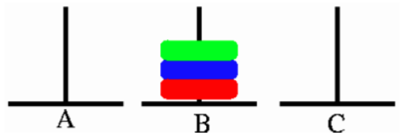
Possible Approach



6. The Tower of Hanoi

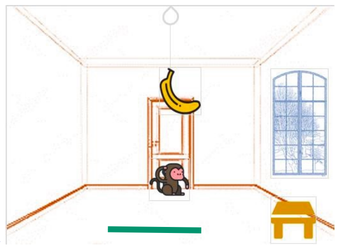
- Somewhere near Hanoi there is a monastery whose monks devote their lives to a very important task.
  - In their courtyard are three tall posts. On these posts is a set of sixty four disks, each with a hole in the center and each of different radius.
  - When the monastery was established, all of the disks were on one of the posts, each disk resting on the one larger than it.
  - The monks task is to move all of the disks to one of the other pegs.
  - Only one disk may be moved at a time, and all the other disks must be on one of the pegs.
  - In addition, at no time during the process may a disk be placed on top of a smaller disk. The third peg can offcourse be used as a temporary resting place for the disks.
- What is the quickest way for the monks to accomplish their mission?
  - Even the best solution to this problem will take the monks a very long time. This is fortunate since legend has it that the world will end when they have finished!





7. The Monkey and Bananas Problem

- A hungry monkey finds himself in a room in which a bunch of bananas is hanging from the ceiling.
  - The monkey, unfortunately cannot reach the bananas.
  - However, in the room there are also a chair and a stick.
  - The ceiling is just the right height so that a monkey standing on a chair could knock the bananas down with the stick.
  - The monkey knows how to move around, carry other things around, reach for the bananas, and wave a stick in the air.
- What is the best sequence of actions for the monkey to take to acquire the lunch?



8. Cryptarithmetic

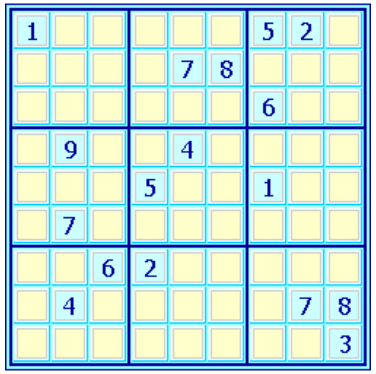
- Consider an arithmetic problem represented by letters, as shown below:

```
    SEND              DONALD
   +MORE            + GERALD
   ----------        --------------
   MONEY              ROBERT
```

  - Assign a decimal digit to each of the letters in such a way that the answer to the problem is correct.
  - If the same letter occurs more than once, it must be assigned the same digit each time.
  - No two different letters may be assigned the same digit.

9.Sudoku

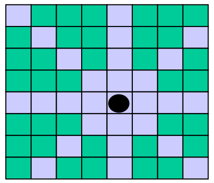
- 9x9 array
- Numbers to be filled with certain rules

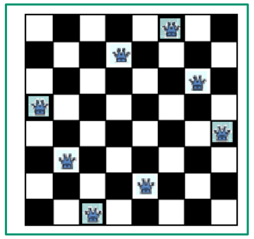


1						5	2	
				7	8			
						6		
	9			4				
			5			1		
	7							
		6	2					
	4						7	8
								3

10. Eight Queen Problem

- It is the problem of putting eight chess queens on an 8×8 chessboard such that none of them is able to capture any other using the standard chess queen's moves.
- The queens must be placed in such a way that no two queens would be able to attack each other. Thus, a solution requires that no two queens share the same row, column, or diagonal.
- The eight queens puzzle is an example of the more general n queens puzzle of placing n queens on an n×n chessboard, where solutions exist only for n = 1 and n ≥ 4.





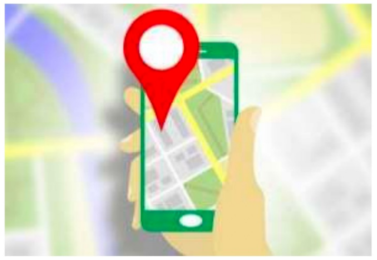
We want

Our aim is to solve all type of problems in the world
- at least our AI algorithm attempt that problem in principal

- Automated Problem solving approach
- Generalized Techniques for
  - Solving Large Classes of Complex Problems
  - if not clear how to start/ proceed :
    - Ask what is input and what is output

AI in Transportation

- **Navigation**



- Google & Waze find the fastest route, by processing traffic data.

- Ride sharing

- Uber & Lyft predict real-time demand using AI techniques, machine learning, deep learning.

AI in Social Media

- **Audience**

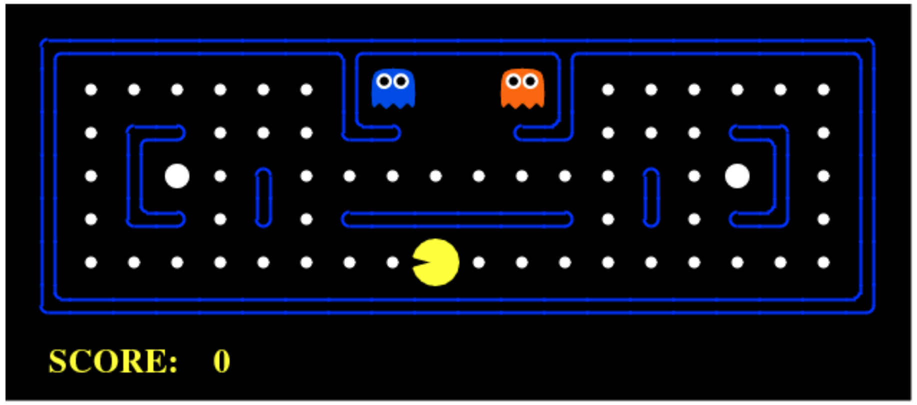
- Facebook & Twitter use AI to decide what content to present in their feeds to different audiences.

- **Content**

- Image recognition and sentiment analysis to ensure that content of the appropriate "mood" is being served.

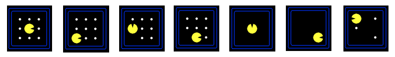
Search Problems

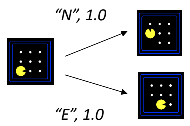
Example : Pac-Man



Problem Representation in AI : **Search Problems**

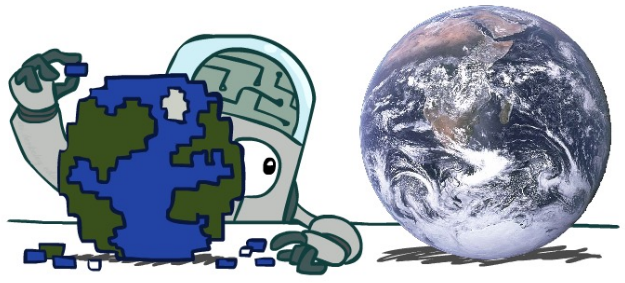
- A search problem consists of:
  - A state space
  - A successor function (with actions, costs)
  - A start state and a goal test





- A solution is a sequence of actions (a plan) which transforms the start state to a goal state

Search Problems Are Models

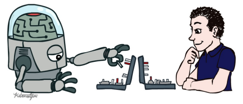


Next :

- Module 2: Automated Problem Solving
  - PART 2.1: Intelligent Agent & Environment
  - PART 2.2: Complex Problems and AI
  - PART 2.3: Problem Solving Methods

References

- Slides adapted from CS188 Instructor: Anca Dragan, University of California, Berkeley
- Slides adapted from CS60045 ARTIFICIAL INTELLIGENCE



(some slides adapted from http://aima.cs.berkeley.edu/)