## 1. General Issues & Overview of AI

### 👉 What is Artificial Intelligence (AI)?

Artificial Intelligence is a branch of computer science that focuses on creating systems capable of performing tasks that normally require human intelligence such as:

- Learning
- Reasoning
- Problem-solving
- Decision making

---

### 🔷 2. AI Problems

AI problems are different from normal programming problems because:

**✓ Characteristics of AI Problems:**

1. **Large search space** (many possible solutions)
2. **Uncertainty** (incomplete or unclear information)
3. **Need for intelligence** (reasoning, learning)
4. **Non-deterministic** (same input may give different outputs)

**👉 Example:**

- Chess playing
- Medical diagnosis
- Speech recognition

---

### 🔷 3. What is an AI Technique?

An AI technique is a method used to solve complex problems efficiently.

**✓ Key Properties:**

1. Uses **knowledge representation**
2. Applies **heuristics (rules of thumb)**
3. Handles **uncertainty**
4. Improves performance over time

👉 **Example:**

- Heuristic search
- Neural networks
- Rule-based systems

---

### 🔷 4. Characteristics of AI Applications

AI applications typically have:

1. **Knowledge-based**
2. **Adaptive (learning capability)**
3. **Handles ambiguity**
4. **Goal-oriented**
5. **Uses reasoning and logic**

👉 **Examples:**

- Chatbots
- Recommendation systems
- Self-driving cars

---

### 🔷 5. Problem Solving in AI

Problem solving means finding a sequence of actions that leads from the **initial state → goal state**.

✓ **Components:**

1. Initial state
2. Goal state
3. Operators (actions)
4. State space

---

### 🔷 6. Search and Control Strategies

AI solves problems mainly using **search techniques**.

✓ **Search Strategy:**

Method used to explore possible solutions.

✓ **Control Strategy:**

Determines:

- Which node to expand
- Order of search

---

🔷 **7. General Problem Solving**

A general problem-solving system works for different types of problems.

✓ **Steps:**

1. Define problem
2. Represent knowledge
3. Apply search strategy
4. Reach solution

---

🔷 **8. Production Systems**

A production system is a rule-based system used in AI.

✓ **Components:**

1. **Rule base** (IF–THEN rules)
2. **Working memory** (current data)
3. **Inference engine** (applies rules)

👉 **Example Rule:**

IF fever AND cough → disease = flu

---

🔷 **9. Control Strategies**

Control strategies decide how rules are applied.

✓ **Types:**

1. Forward chaining

2. Backward chaining

---

### 🔷 10. Forward Chaining

**👉 Data-driven approach**

- Starts from known facts
- Applies rules to reach conclusion

**✓ Steps:**

1. Start with initial facts
2. Apply rules
3. Generate new facts
4. Continue until goal is reached

**👉 Example:**

Symptoms → Diagnosis

---

### 🔷 11. Backward Chaining

**👉 Goal-driven approach**

- Starts from goal
- Works backward to find supporting facts

**✓ Steps:**

1. Start with goal
2. Check rules that support goal
3. Verify required conditions

**👉 Example:**

Diagnosis → Check symptoms

---

### 🔷 12. Exhaustive Search

Exhaustive search means checking **all possible solutions**.

**✓ Advantage:**

- Guaranteed solution

❌ **Disadvantage:**

- Very slow
- High computation cost

---

🔷 **13. Depth First Search (DFS)**

👉 **Strategy:**

- Explore deepest node first

✓ **Working:**

1. Go deep into one branch
2. Backtrack if no solution

✓ **Advantages:**

- Less memory
- Simple

❌ **Disadvantages:**

- May get stuck in infinite path
- Not optimal

---

🔷 **14. Breadth First Search (BFS)**

👉 **Strategy:**

- Explore level by level

✓ **Working:**

1. Visit all nodes at current level
2. Move to next level

✓ **Advantages:**

- Finds shortest path
- Complete

❌ **Disadvantages:**

- High memory usage

---

🔚 **Summary Table**

| Topic | Key Idea |
|---|---|
| AI | Machine intelligence |
| AI Technique | Efficient problem solving method |
| Problem Solving | Initial → Goal state |
| Production System | Rule-based system |
| Forward Chaining | Data-driven |
| Backward Chaining | Goal-driven |
| DFS | Deep search |
| BFS | Level-wise search |



Unit-2.0: Heuristic Search Techniques 9 hrs Hill climbing; Branch and Bound technique; Best first search and A* algorithm; AND/OR Graphs; Problem reduction and AO* algorithm; Constraint Satisfaction problems Game Playing Min Max Search procedure; Alpha-Beta cutoff; Additional Refinements.

**1. Heuristic Search Techniques**

👉 **What is Heuristic?**

A **heuristic** is a rule or method used to make problem solving faster by using **experience or estimation**.

- Instead of checking all possibilities (like BFS/DFS), heuristic search uses **intelligence**.
- It reduces **time and complexity**.

✓ **Example:**

In maps → choosing shortest path using distance estimate.

---

🔷 **2. Hill Climbing Algorithm**

👉 **Idea:**

- Always move toward the **best (highest value) neighbor**
- Like climbing a hill to reach the peak

✓ **Steps:**

1. Start from initial state
2. Evaluate neighbors
3. Move to best neighbor
4. Repeat until no better state

✓ **Advantages:**

- Simple
- Fast

❌ **Disadvantages:**

- **Local maxima problem**
- Plateau problem
- May not find global optimum

---

🔷 **3. Branch and Bound Technique**

👉 **Idea:**

- Systematically explore solutions
- Keep track of **best solution (bound)**

✓ **Working:**

1. Divide problem into subproblems (branching)
2. Calculate cost (bound)
3. Ignore paths worse than best solution

✓ **Used in:**

- Optimization problems
- Travelling Salesman Problem (TSP)

---

### 🔷 4. Best First Search

👉 **Idea:**

- Expand node with **best heuristic value (h(n))**

✓ **Formula:**

Choose node with **minimum h(n)**

✓ **Features:**

- Uses priority queue
- Faster than BFS/DFS

❌ **Problem:**

- Not always optimal

---

### 🔷 5. A* Algorithm

👉 **Most important algorithm (very common in exams)**

✓ **Formula:**

$$f(n) = g(n) + h(n)$$

Where:

- **g(n)** = cost from start to current node
- **h(n)** = estimated cost to goal
- **f(n)** = total estimated cost

✓ **Properties:**

- Complete ✓
- Optimal ✓ (if h(n) is admissible)

✓ **Advantage:**

- Combines **best-first + uniform cost search**

---

### 🔷 6. AND/OR Graphs

👉 **Used for complex problem decomposition**

✓ **Types:**

1. **OR Node** → choose one path
2. **AND Node** → must satisfy all paths

👉 **Example:**

- OR: choose one route
- AND: complete all sub-tasks

---

🔷 **7. Problem Reduction**

👉 **Idea:**

Break a big problem into **smaller subproblems**

✓ **Used in:**

- Planning
- Theorem proving

---

🔷 **8. AO* Algorithm**

👉 **Extension of A* for AND/OR graphs**

✓ **Working:**

1. Expand most promising node
2. Consider AND/OR conditions
3. Update cost recursively

✓ **Advantage:**

- Efficient for structured problems

---

🔷 **9. Constraint Satisfaction Problems (CSP)**

👉 **Definition:**

A problem where solution must satisfy **constraints**

✓ **Components:**

1. Variables

2. Domains (possible values)
3. Constraints

👉 **Example:**

- Sudoku
- Map coloring

✓ **Techniques:**

- Backtracking
- Constraint propagation

---

🔷 **10. Game Playing in AI**

AI is used in games like:

- Chess
- Tic-tac-toe

✓ **Goal:**

- Choose best move against opponent

---

🔷 **11. Min-Max (Minimax) Algorithm**

👉 **Idea:**

- Two players:
  - MAX (AI) → maximize score
  - MIN (opponent) → minimize score

✓ **Working:**

1. Generate game tree
2. Apply min/max alternately
3. Choose optimal move

---

🔷 **12. Alpha-Beta Pruning**

👉 **Optimization of Minimax**

✓ **Idea:**

- Remove branches that **cannot affect final result**

✓ **Terms:**

- **Alpha (α)** → best value for MAX
- **Beta (β)** → best value for MIN

✓ **Advantage:**

- Reduces search space
- Faster than minimax

---

### 🔷 13. Additional Refinements

These improve performance:

✓ **Techniques:**

1. **Move ordering** → check best moves first
2. **Iterative deepening**
3. **Heuristic evaluation functions**
4. **Transposition tables** (store results)

---

### 🔚 Quick Summary Table

| Topic | Key Idea |
|---|---|
| Heuristic | Smart shortcut |
| Hill Climbing | Local optimization |
| Branch & Bound | Optimal solution tracking |
| Best First | Expand best node |
| A* | g(n) + h(n) |
| AND/OR Graph | Complex decisions |
| AO* | A* for AND/OR |

| Topic | Key Idea |
|---|---|
| CSP | Constraints-based problems |
| Minimax | Game strategy |
| Alpha-Beta | Pruning |

### 1. Knowledge Representation (KR)

👉 **Definition:**

Knowledge Representation is the method of **representing information about the world** in a form that a computer system can use to solve problems.

✓ **Purpose:**

- Store knowledge
- Reason about problems
- Make decisions

✓ **Types:**

- Logical representation
- Semantic networks
- Production rules

---

### 🔷 2. First Order Predicate Calculus (FOPC)

👉 **Definition:**

A formal system used to represent **facts, objects, and relationships**.

✓ **Components:**

1. **Constants** → specific objects
    - Example: Ram, Delhi
2. **Variables** → general objects
    - Example: x, y
3. **Predicates** → relationships

- Example: Loves(Ram, Sita)

4. **Functions** → mapping
5. **Quantifiers**:
    - ∀ (For all)
    - ∃ (There exists)

---

✓ **Example:**

"All humans are mortal"

→ ∀x (Human(x) → Mortal(x))

---

🔷 **3. Skolemization**

👉 **Definition:**

Process of removing **existential quantifiers (∃)** from logical expressions.

✓ **Purpose:**

- Convert formula into standard form (Clause form)

✓ **Rules:**

1. Replace ∃ variable with **Skolem constant** (if no ∀ before it)
2. Replace ∃ variable with **Skolem function** (if inside ∀)

---

✓ **Example:**

∃x Loves(x, Ram)
→ Loves(A, Ram) (A = Skolem constant)

---

🔷 **4. Resolution Principle**

👉 **Definition:**

A rule of inference used to **prove statements logically**.

✓ **Idea:**

- Convert statements into clauses

- Apply resolution to derive contradiction

---

✓ **Example:**

Clause 1: A ∨ B
Clause 2: ¬A

→ Result: B

---

✓ **Use:**

- Automated theorem proving
- Logic programming

---

🔷 **5. Unification**

👉 **Definition:**

Process of making two logical expressions **identical** by substituting variables.

✓ **Example:**

Loves(x, Ram)
Loves(Shyam, Ram)

→ x = Shyam

---

✓ **Importance:**

- Required in resolution
- Helps match patterns

---

🔷 **6. Inference Mechanisms**

👉 **Definition:**

Techniques used to **derive new knowledge from existing knowledge**

✓ **Types:**

1. Forward chaining
2. Backward chaining

3. Resolution

---

### 🔷 7. Horn Clauses

**👉 Definition:**

A special type of logical expression with **at most one positive literal**

---

**✓ Form:**

$A \leftarrow B_1 \wedge B_2 \wedge B_3$

(If B1 AND B2 THEN A)

---

**✓ Example:**

Mortal(x) ← Human(x)

---

**✓ Importance:**

- Used in **Prolog programming**
- Efficient reasoning

---

### 🔷 8. Semantic Networks

**👉 Definition:**

Graph-based representation of knowledge

**✓ Components:**

1. **Nodes** → objects/concepts
2. **Edges** → relationships

---

**✓ Example:**

- Dog → is-a → Animal
- Dog → has → Tail

---

✓ **Advantages:**

- Easy to visualize
- Represents relationships clearly

❌ **Disadvantages:**

- Not good for complex logic

---

🔚 **Summary Table**

| Topic | Key Idea |
|---|---|
| KR | Represent knowledge |
| FOPC | Logic-based representation |
| Skolemization | Remove ∃ |
| Resolution | Prove using contradiction |
| Unification | Match expressions |
| Inference | Derive new knowledge |
| Horn Clause | Simple rule format |
| Semantic Network | Graph representation |

**1. Frame Systems**

👉 **Definition:**

A **frame** is a data structure used to represent knowledge in AI, similar to an object in programming.

✓ **Structure:**

A frame consists of:

- **Slots** → attributes
- **Values** → data in slots

---

✓ **Example:**

Frame: **Car**

| Slot | Value |
|---|---|
| Type | Vehicle |
| Wheels | 4 |
| Fuel | Petrol |

---

✓ **Features:**

- Organized knowledge
- Supports inheritance
- Easy to update

---

🔷 **2. Value Inheritance**

👉 **Definition:**

Frames can inherit values from **parent frames**

---

✓ **Example:**

Frame: **Vehicle**

- Wheels = 4

Frame: **Car (child of Vehicle)**

- Inherits Wheels = 4

---

✓ **Advantage:**

- Avoids repetition
- Efficient knowledge storage

---

🔷 **3. Scripts**

👉 **Definition:**

Scripts represent **typical sequences of events** in a situation.

---

✓ **Example: Restaurant Script**

1. Enter restaurant
2. Order food
3. Eat
4. Pay bill
5. Leave

---

✓ **Use:**

- Natural Language Processing (NLP)
- Understanding stories

---

🔷 **4. Conceptual Dependency (CD)**

👉 **Definition:**

A technique to represent meaning of sentences **independent of language**

---

✓ **Idea:**

- Focus on **actions and relationships**
- Remove ambiguity

---

✓ **Example:**

“Ram gave a book to Shyam”

→ Represented using actions like:

- Transfer
- Object
- Actor

---

✓ **Advantage:**

- Language-independent
- Useful in NLP

---

🔷 **5. AI Programming Languages**

AI uses special languages designed for:

- Symbolic processing
- Logic-based reasoning

---

🔷 **6. Introduction to LISP**

👉 **Full Form:**

**LISt Processing Language**

✓ **Features:**

- Oldest AI language
- Works with lists
- Uses prefix notation

---

✓ **Syntax Example:**

(+ 2 3)

→ Output: 5

---

🔷 **7. Numeric Functions in LISP**

✓ **Examples:**

(+ 2 3) → 5
(- 5 2) → 3
(* 2 3) → 6
(/ 6 2) → 3

---

## 🔷 8. List Manipulation Functions

**✓ Important Functions:**

| Function | Meaning |
|---|---|
| CAR | First element |
| CDR | Rest of list |
| CONS | Add element |

---

**✓ Example:**

(CAR '(1 2 3)) → 1
(CDR '(1 2 3)) → (2 3)

---

## 🔷 9. Iteration and Recursion in LISP

**✓ Iteration:**

- Looping using functions

**✓ Recursion:**

- Function calling itself

---

**✓ Example (Recursion):**

Factorial:

```
(defun fact (n)
  (if (= n 0)
     1
     (* n (fact (- n 1)))))
```

---

## 🔷 10. Property List & Arrays

**✓ Property List:**

- Stores key-value pairs

Example:

(Name Ram Age 20)

---

✓ **Arrays:**

- Store multiple values in indexed form
- Similar to other languages

---

🔷 **11. Introduction to PROLOG**

👉 **Full Form:**

**Programming in Logic**

✓ **Features:**

- Logic-based language
- Uses facts and rules
- Based on predicate logic

---

✓ **Example:**

father(ram, shyam).

Rule:

grandfather(X, Z) :- father(X, Y), father(Y, Z).

---

✓ **Key Concepts:**

- Facts
- Rules
- Queries

---

🔚 **Summary Table**

| **Topic** | **Key Idea** |
|---|---|
| Frame | Structured knowledge |
| Inheritance | Share values |

| Topic | Key Idea |
|---|---|
| Script | Sequence of events |
| CD | Meaning representation |
| LISP | List processing |
| List Functions | CAR, CDR, CONS |
| Recursion | Self-calling function |
| PROLOG | Logic programming |

**1. Natural Language Processing (NLP)**

👉 **Definition:**

NLP is a branch of AI that enables computers to **understand, interpret, and generate human language**.

✓ **Applications:**

- Chatbots
- Machine translation
- Speech recognition

---

🔷 **2. Context-Free Grammar (CFG)**

👉 **Definition:**

A grammar used to describe the **structure of sentences**.

✓ **Components:**

1. **Non-terminals** (S, NP, VP)
2. **Terminals** (words)
3. **Production rules**
4. **Start symbol**

---

✓ **Example:**

S → NP VP
NP → Ram
VP → eats mango

---

✓ **Use:**

- Syntax analysis
- Sentence parsing

---

🔷 **3. Recursive Transition Networks (RTN)**

👉 **Definition:**

A **graph-based representation of grammar**.

✓ **Features:**

- Nodes → states
- Arcs → transitions
- Supports recursion

---

✓ **Advantage:**

- Simple parsing
- Visual representation

---

🔷 **4. Augmented Transition Networks (ATN)**

👉 **Definition:**

An extension of RTN with **extra features like memory and conditions**

---

✓ **Features:**

- Registers (store values)
- Conditions (tests)
- Actions

---

✓ **Advantage:**

- More powerful than RTN
- Handles complex sentences

---

🔷 **5. Semantic Analysis**

👉 **Definition:**

Understanding the **meaning of a sentence**

---

✓ **Tasks:**

- Word sense disambiguation
- Relationship identification
- Logical interpretation

---

🔷 **6. Case Grammar**

👉 **Definition:**

Focuses on **roles of words in a sentence**

---

✓ **Common Cases:**

- Agent (doer)
- Object
- Instrument

---

✓ **Example:**

“Ram cut apple with knife”

- Agent → Ram
- Object → Apple
- Instrument → Knife

---

### 🔷 7. Logic Grammar

**👉 Definition:**

Uses **logical representation (predicate logic)** for language

---

**✓ Example:**

Ram eats mango
→ Eats(Ram, Mango)

---

---

### 🔷 8. Planning in AI

**👉 Definition:**

Planning is the process of **finding a sequence of actions to achieve a goal**

---

### 🔷 9. Blocks World (Example Domain)

**👉 Description:**

- Blocks placed on a table
- Goal: rearrange blocks

---

**✓ Example:**

Initial: A on B
Goal: B on A

---

**✓ Used for:**

- Testing planning algorithms

---

### 🔷 10. Components of Planning System

1. **Initial state**

2. **Goal state**
3. **Operators (actions)**
4. **Plan (sequence of steps)**

---

### 🔷 11. Goal Stack Planning (Linear Planning)

**👉 Idea:**

- Use a **stack of goals and sub-goals**

---

**✓ Steps:**

1. Put goal on stack
2. Break into sub-goals
3. Solve step by step

---

**✓ Feature:**

- Linear execution (one step at a time)

---

### 🔷 12. Non-Linear Planning (Constraint Posting)

**👉 Idea:**

- Actions are not strictly ordered
- Use constraints instead

---

**✓ Advantage:**

- Flexible
- More efficient

---

### 🔷 13. Probabilistic Reasoning & Uncertainty

**👉 Definition:**

Handling **uncertain or incomplete information**

---

✓ **Example:**

Medical diagnosis with probability

---

🔷 **14. Probability Theory**

👉 **Formula:**

$$P(A) = \frac{\text{Number of favorable outcomes}}{\text{Total outcomes}}$$

---

✓ **Concepts:**

- Events
- Conditional probability
- Independence

---

🔷 **15. Bayes Theorem**

👉 **Very important (frequent exam question)**

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

$$P(A)$$

$$P(B \mid A)$$

$$P(B \mid \neg A)$$

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)} \approx 0.68,\ P(B) \approx 0.25$$

P(B)=0.25P(B|A)P(A)=0.17P(A|B)~0.68Posterior = useful evidence / total evidence

---

✓ **Use:**

- Medical diagnosis
- Spam filtering

---

## 🔷 16. Bayesian Networks

👉 **Definition:**

A **graphical model** representing probabilistic relationships

---

✓ **Features:**

- Nodes → variables
- Edges → dependencies
- Directed acyclic graph (DAG)

---

✓ **Example:**

Rain → Wet ground

---

## 🔷 17. Certainty Factor (CF)

👉 **Definition:**

Measure of **belief in a hypothesis**

---

✓ **Range:**

$-1 \leq CF \leq 1$

- 1 → completely true
- 0 → unknown
- -1 → false

---

✓ **Formula:**

$CF = MB - MD$

Where:

- MB = Measure of belief
- MD = Measure of disbelief

---

🔚 **Summary Table**

| Topic | Key Idea |
|---|---|
| NLP | Language understanding |
| CFG | Sentence structure |
| RTN | Graph grammar |
| ATN | Advanced RTN |
| Semantic Analysis | Meaning |
| Case Grammar | Roles |
| Planning | Action sequence |
| Blocks World | Example domain |
| Goal Stack | Linear planning |
| Non-linear | Flexible planning |
| Probability | Uncertainty handling |
| Bayes | Conditional probability |
| Bayesian Network | Graph model |
| Certainty Factor | Degree of belief |



**1. Introduction to Expert Systems**

👉 **Definition:**

An **Expert System (ES)** is an AI program that mimics the **decision-making ability of a human expert**.

---

✓ **Features:**

- Uses **knowledge base**
- Provides **solutions/advice**
- Can explain reasoning

- Works in specific domain

---

✓ **Examples:**

- Medical diagnosis
- Financial decision systems

---

🔷 **2. Architecture of Expert Systems**

👉 **Main Components:**

1. **Knowledge Base**
    - Stores facts and rules
2. **Inference Engine**
    - Applies rules to solve problems
3. **User Interface**
    - Interaction with user
4. **Explanation System**
    - Explains reasoning
5. **Working Memory**
    - Stores current data

---

✓ **Diagram Idea (for exam):**

User → Interface → Inference Engine → Knowledge Base
↓
Explanation System

---

🔷 **3. Expert System Shells**

👉 **Definition:**

A **shell** is an expert system **without knowledge base**

---

✓ **Features:**

- Provides framework
- User adds domain knowledge

---

✓ **Examples:**

- EMYCIN
- CLIPS

---

🔷 **4. Knowledge Acquisition**

👉 **Definition:**

Process of **collecting knowledge from experts**

---

✓ **Methods:**

1. Interviews
2. Observation
3. Data analysis
4. Machine learning

---

❗ **Problem:**

- Experts may not explain knowledge clearly
- Time-consuming

---

🔷 **5. Case Study: MYCIN**

👉 **Definition:**

**MYCIN** is a famous expert system for **medical diagnosis (bacterial infections)**

---

✓ **Features:**

- Used rule-based reasoning
- Used **certainty factors (CF)**

- Suggested antibiotics

---

✓ **Example Rule:**

IF infection is bacterial
THEN prescribe antibiotic

---

✓ **Importance:**

- One of the earliest successful expert systems

---

🔷 **6. Learning in AI**

👉 **Definition:**

Learning is the ability of a system to **improve performance using experience**

---

🔷 **7. Rote Learning**

👉 **Definition:**

Learning by **memorization**

---

✓ **Example:**

- Storing solutions directly

---

✓ **Advantage:**

- Fast retrieval

❌ **Disadvantage:**

- No understanding

---

🔷 **8. Learning by Induction**

👉 **Definition:**

Learning **general rules from examples**

---

✓ **Example:**

Given:

- Dog is animal
- Cat is animal

→ Learn: All pets are animals

---

✓ **Use:**

- Machine learning
- Pattern recognition

---

🔷 **9. Explanation-Based Learning (EBL)**

👉 **Definition:**

Learning by **understanding why something works**

---

✓ **Steps:**

1. Solve a problem
2. Explain reasoning
3. Generalize solution

---

✓ **Example:**

- Learn concept after solving math problem

---

✓ **Advantage:**

- Deep understanding
- Efficient learning

---