CS 156:Introduction to Artificial Intelligence

Instructor: Dr. Sayma Akther

San José State University

Comparative Overview of AI Search Strategies

Category	Definition & Key Components	Examples/Key Algorithms	Usage
Uninformed Search	Search strategies that explore the search space without any information about the problem other than its definition.	Breadth-First Search (BFS), Depth-First Search (DFS), Uniform Cost Search	Suitable for problems where the solution path is unknown.
Informed Search	Search strategies that use knowledge about the problem to find solutions more efficiently. Heuristics are functions that estimate how close a state is to a solution.	Greedy Best-First Search, A* Search	Efficient for problems with some insights or knowledge about the solution space.
Game Playing	Deals with environments where an agent's action is countered by one or more opposing agents. Competitive in nature.	-	For competitive environments with agents aiming to maximize their win chances.
Adversarial Search	A strategy that considers the actions of opposing agents (adversaries) when deciding on the best move.	Minimax, Alpha-Beta Pruning	Mainly used in two-player games where agents take turns making moves.

Another Comparative Summary

Category	Key Characteristics	Advantages	Disadvantages	Common Use Cases
Uninformed Search	No prior knowledge about the solution path.	Simple; Often exhaustive, ensuring solution found.	Can be inefficient; Might explore irrelevant paths.	Maze-solving, puzzle games.
Informed Search	Uses heuristics to guide search.	Faster; More efficient; Can be more optimal.	Dependent on heuristic quality.	Route planning, scheduling.
Game Playing	Competitive; Multi-agent environment.	Finds best moves considering the opponent.	Computationally expensive for complex games.	Chess, Go, Tic-tac-toe.
Adversarial Search	Assumes actions of opponents to decide the best move.	Prunes unnecessary moves; Often optimal.	Requires good evaluation functions.	Board games with turn-based strategies.

Constraint Satisfaction Problems (CSPs)

- **Constraint Satisfaction Problems (CSPs)** are mathematical problems defined as a set of objects whose state must satisfy several constraints or limitations.
- **Examples:** Sudoku, map coloring, and the eight-queen puzzle.

Components of a CSP

- A **CSP** involves a set of variables, a domain of values for each variable, and a set of constraints restricting the values the variables can take.
  1. **Variables**: Finite set of variables $X_1, X_2, \ldots, X_n$
  2. **Domains:** Nonempty domain of possible values for each variable D1, D2, …, Dn
     1. Each Di corresponds to the set of possible values that Xi can take.
  3. **Constraints**: Specify allowable combinations of values.
     - Finite set of constraints $C_1, C_2, \ldots, C_m$
       - Each constraint C_i limits the values that variables can take,
       - e.g., $X_1 \neq X_2$
     - Each constraint C_i is a pair <scope, relation>
       - Scope = Tuple of variables that participate in the constraint.
       - Relation = List of allowed combinations of variable values.

Types of Constraints

1. **Unary Constraints**: Restrict the value of a single variable.
   - **Example:** X1X1 can only be a prime number.
2. **Binary Constraints**: Relate two variables.
   - **Example:** X1X1 ≠ X2X2.
3. **Higher-Order Constraints**: Involve three or more variables.

Backtracking Search for CSPs

1. It's a depth-first search with one variable assigned per level of the tree.
   - A recursive algorithm where for each variable, it tries each value in its domain and checks if it satisfies the constraints.
   - If no violation, it moves to the next variable.
2. If no assignment is possible for a variable, go back (backtrack) to the previous level.
   - If a violation is found, it "backtracks" and tries the next value in the domain.
3. Can be enhanced with various strategies like forward checking.

Forward Checking

- Once a variable X is assigned, the forward-checking process checks all unassigned variables that are connected to X by a constraint and prunes from their domains any values that violate the constraint.

Real-World Applications of CSPs

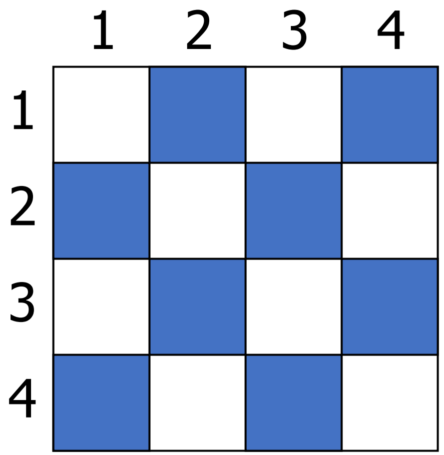
1. **Timetabling problems**: Scheduling university courses without classroom and time clashes.
2. **Map coloring problems**: Assigning colors to neighboring regions on a map such that no two neighboring regions have the same color.
3. **Job scheduling**: Assigning jobs to machines, ensuring efficient usage.
4. **The queen puzzle**
5. **Sudoku**

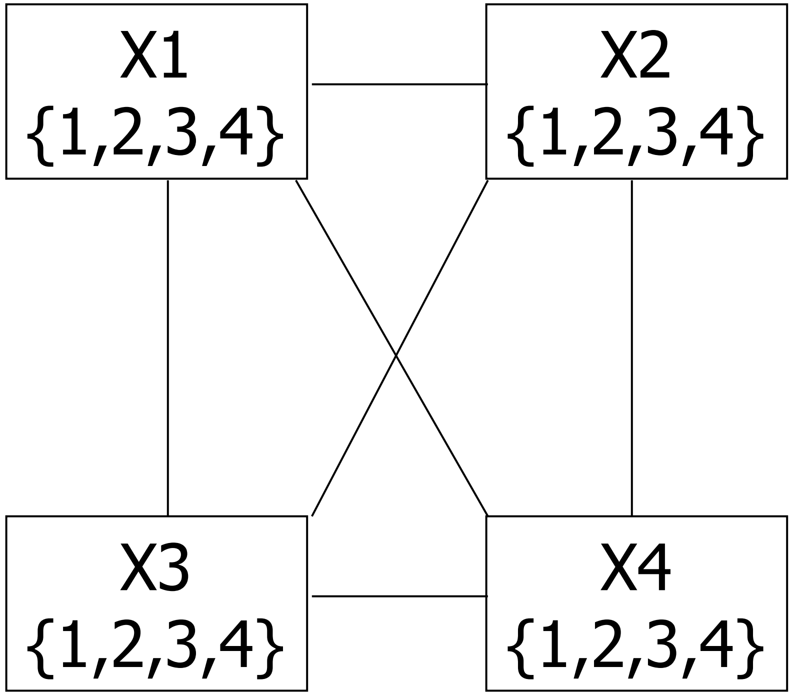
Sudoku as a Constraint Satisfaction Problem (CSP)

- Variables: 81 variables
  - A1, A2, A3, …, I7, I8, I9
  - Letters index rows, top to bottom
  - Digits index columns, left to right
- Domains: The nine positive digits
  - A1 $\in$ {1, 2, 3, 4, 5, 6, 7, 8, 9}
  - Etc.
- Constraints: 27 *Alldiff* constraints
  - *Alldiff*(A1, A2, A3, A4, A5, A6, A7, A8, A9)
  - Etc.

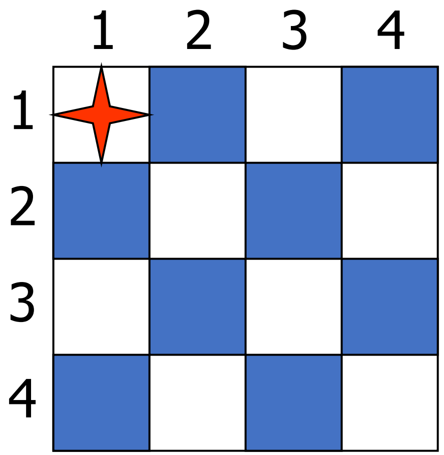
2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

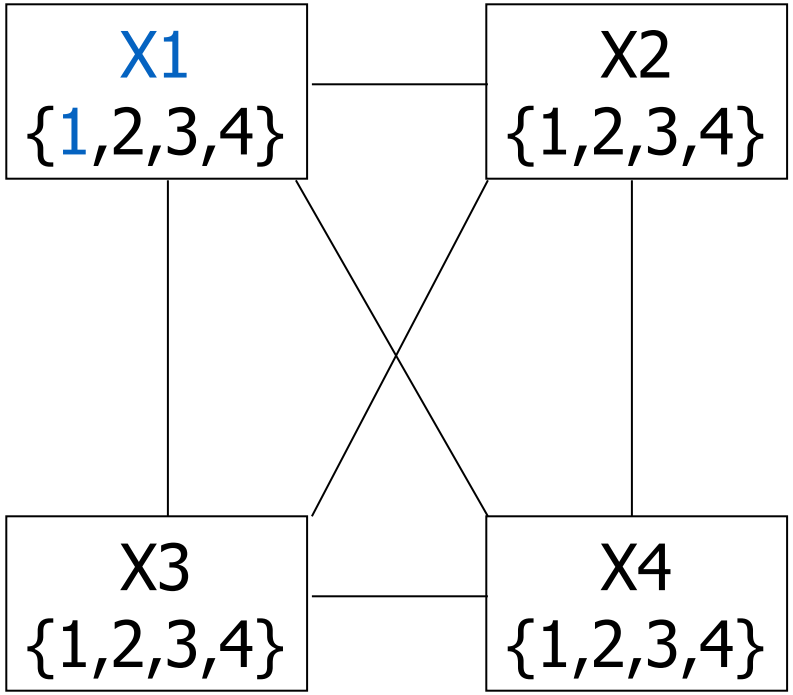
Example: 4-Queens Problem



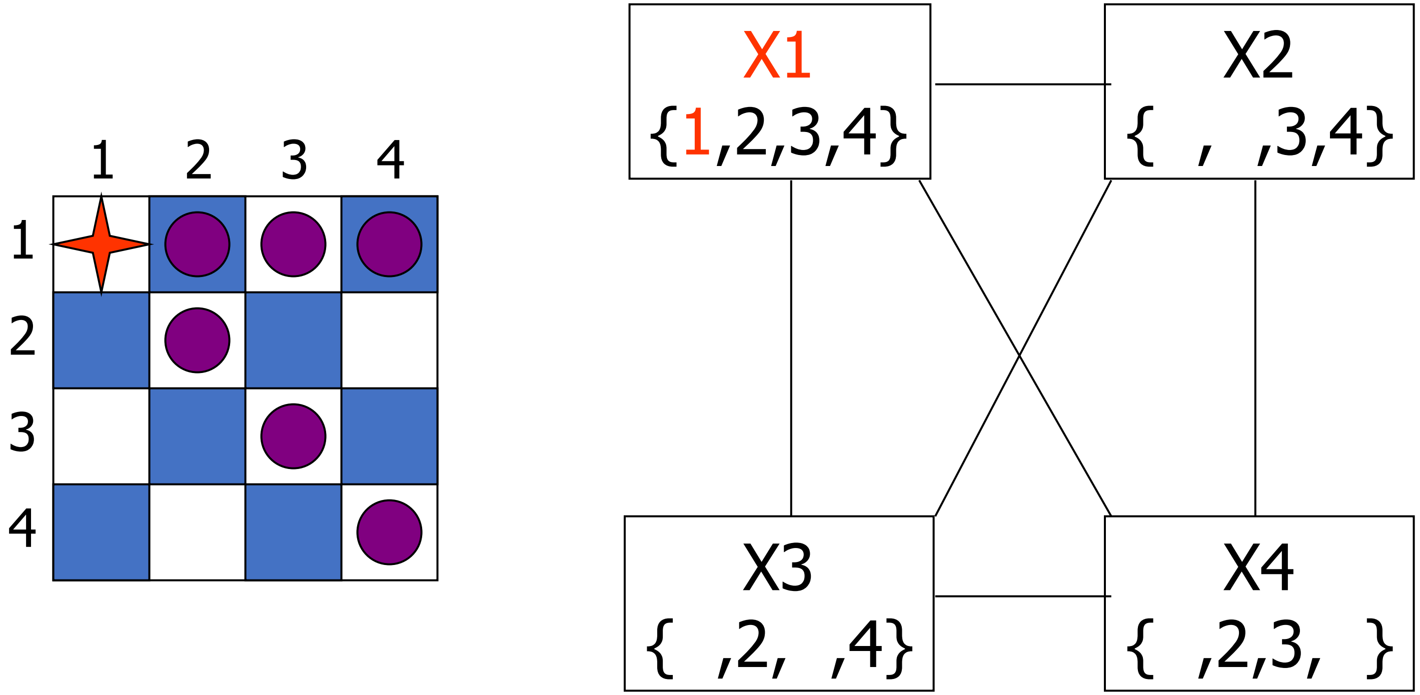


Example: 4-Queens Problem

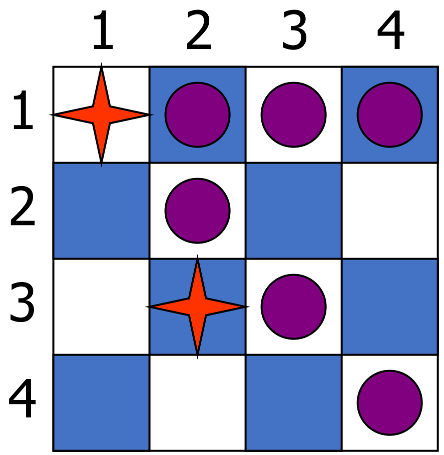


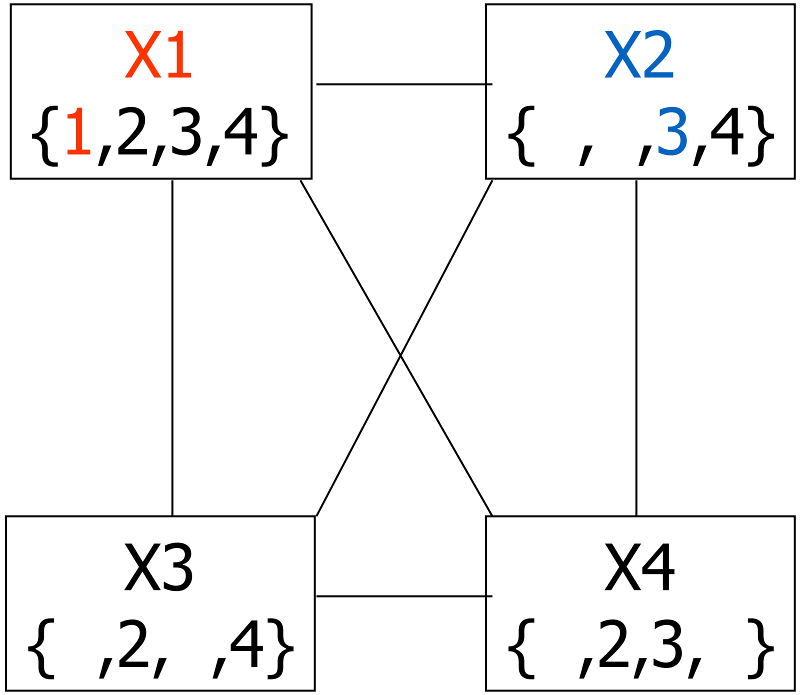


Example: 4-Queens Problem

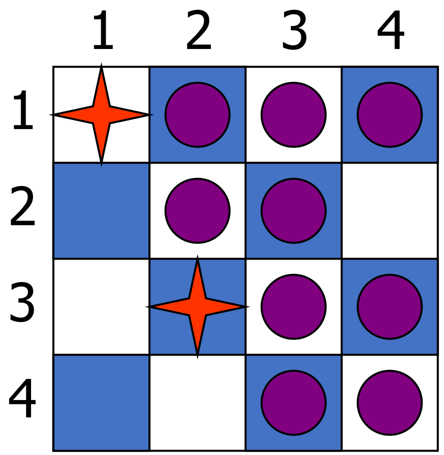


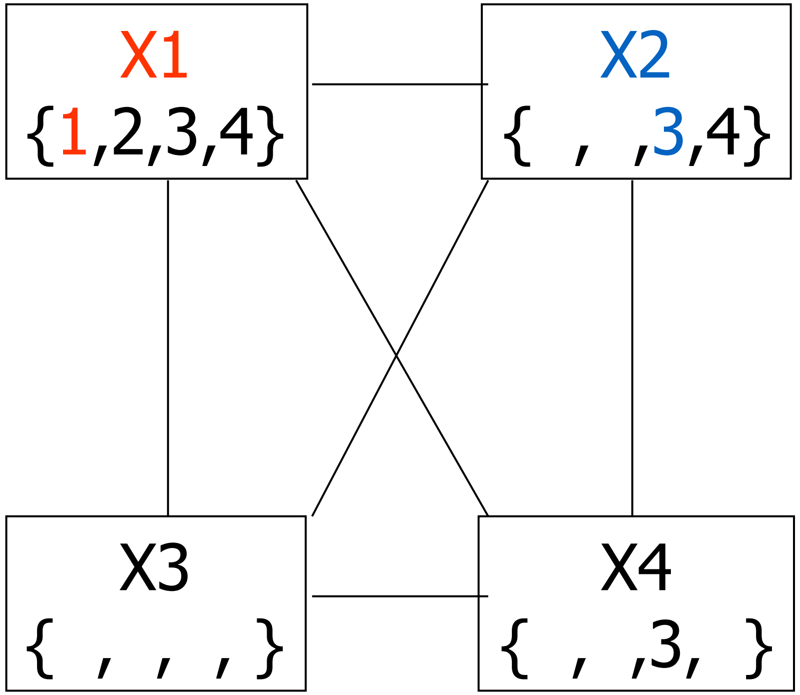
Example: 4-Queens Problem



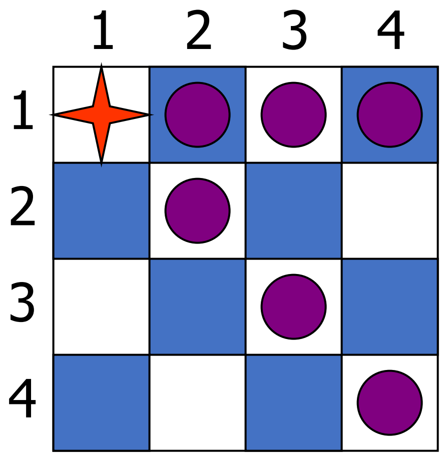


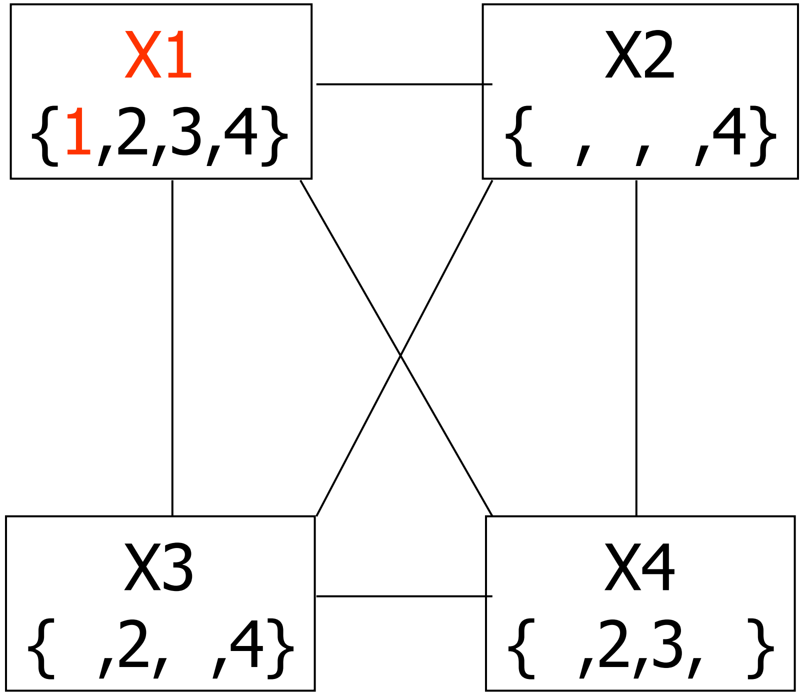
Example: 4-Queens Problem



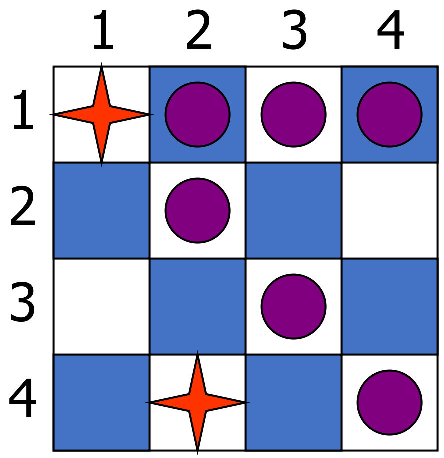


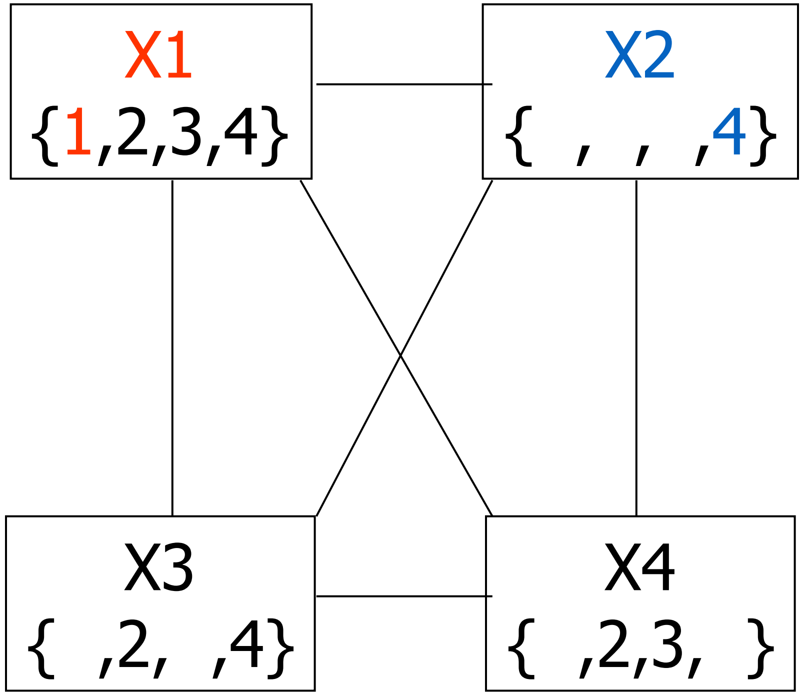
Example: 4-Queens Problem



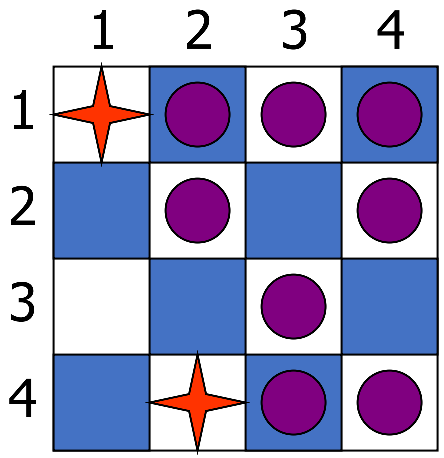


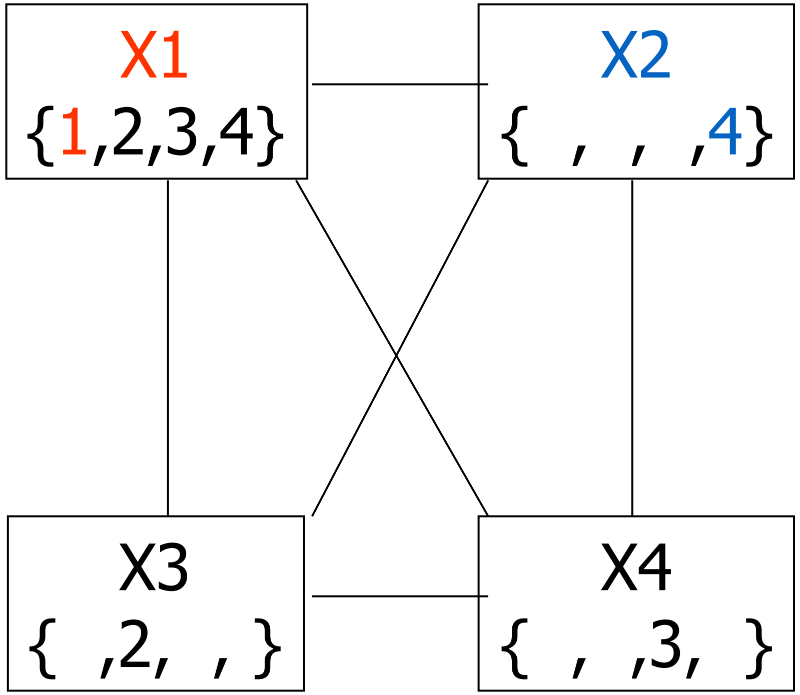
Example: 4-Queens Problem



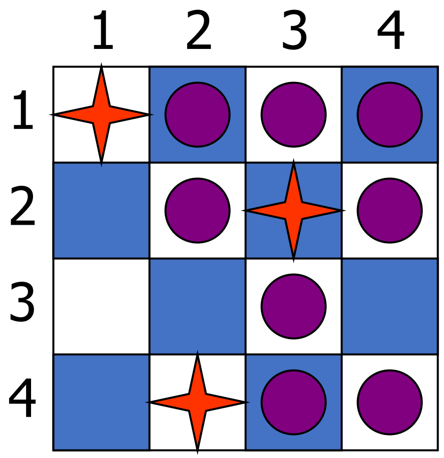


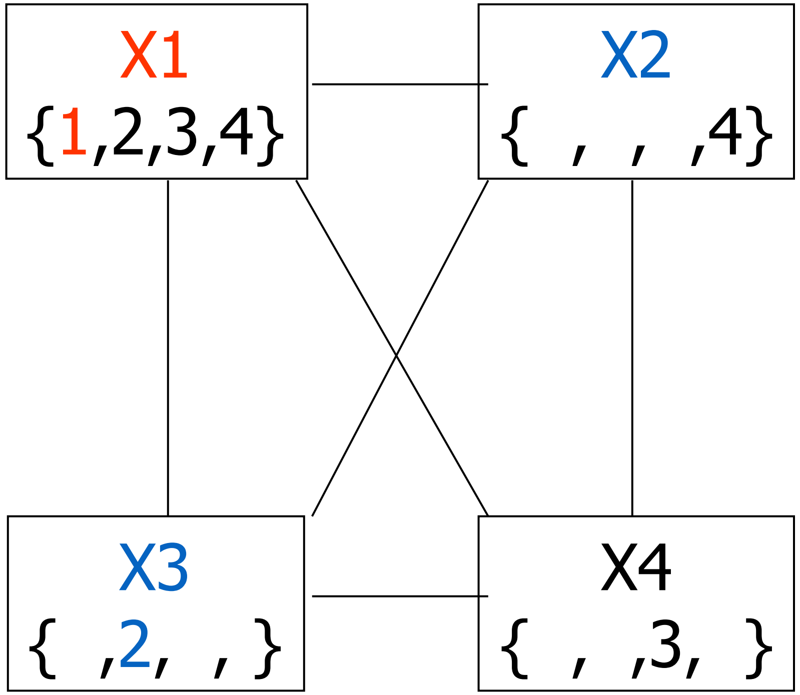
Example: 4-Queens Problem



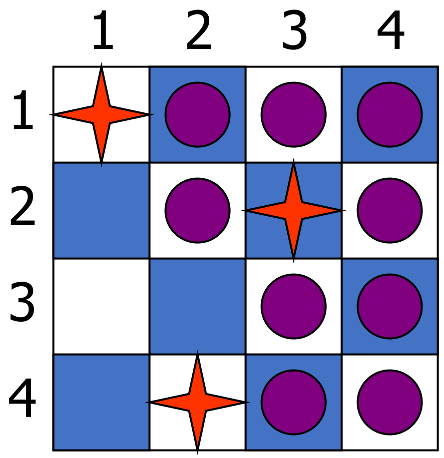


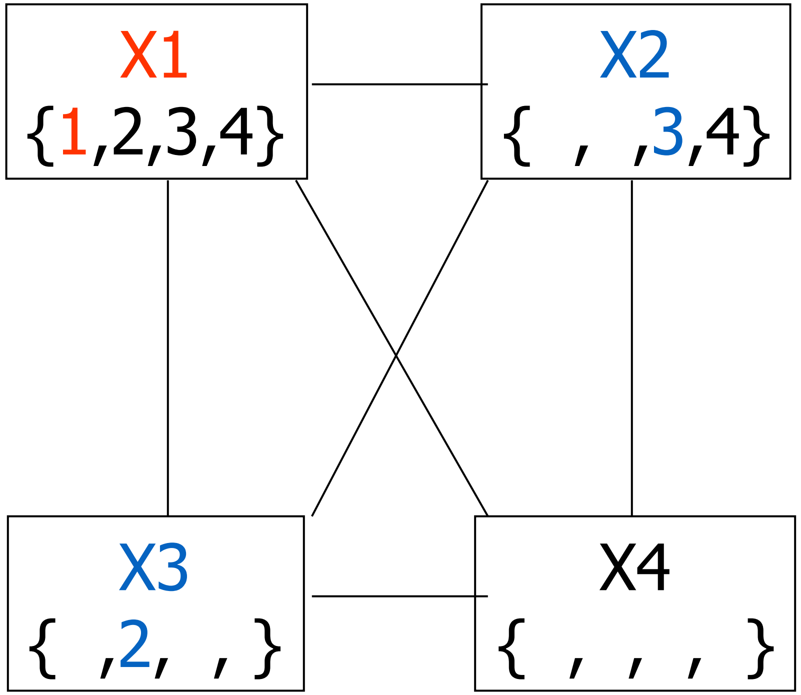
Example: 4-Queens Problem



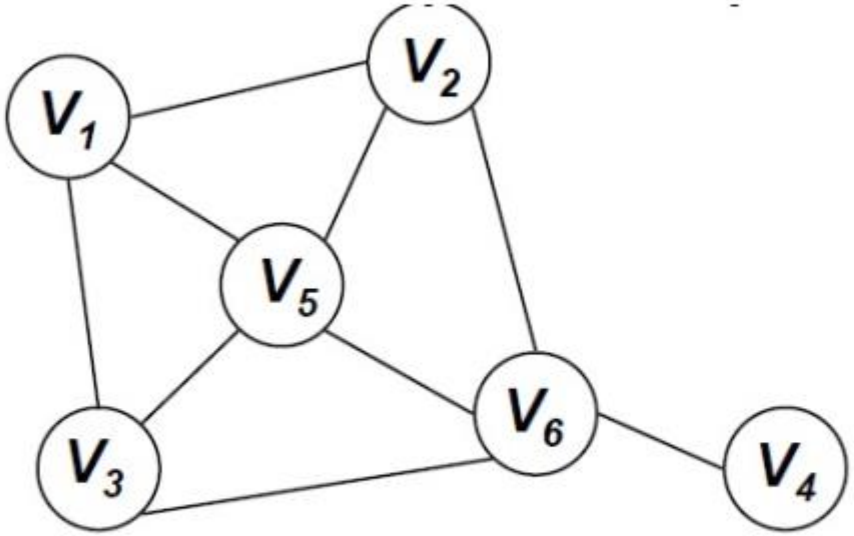


Example: 4-Queens Problem



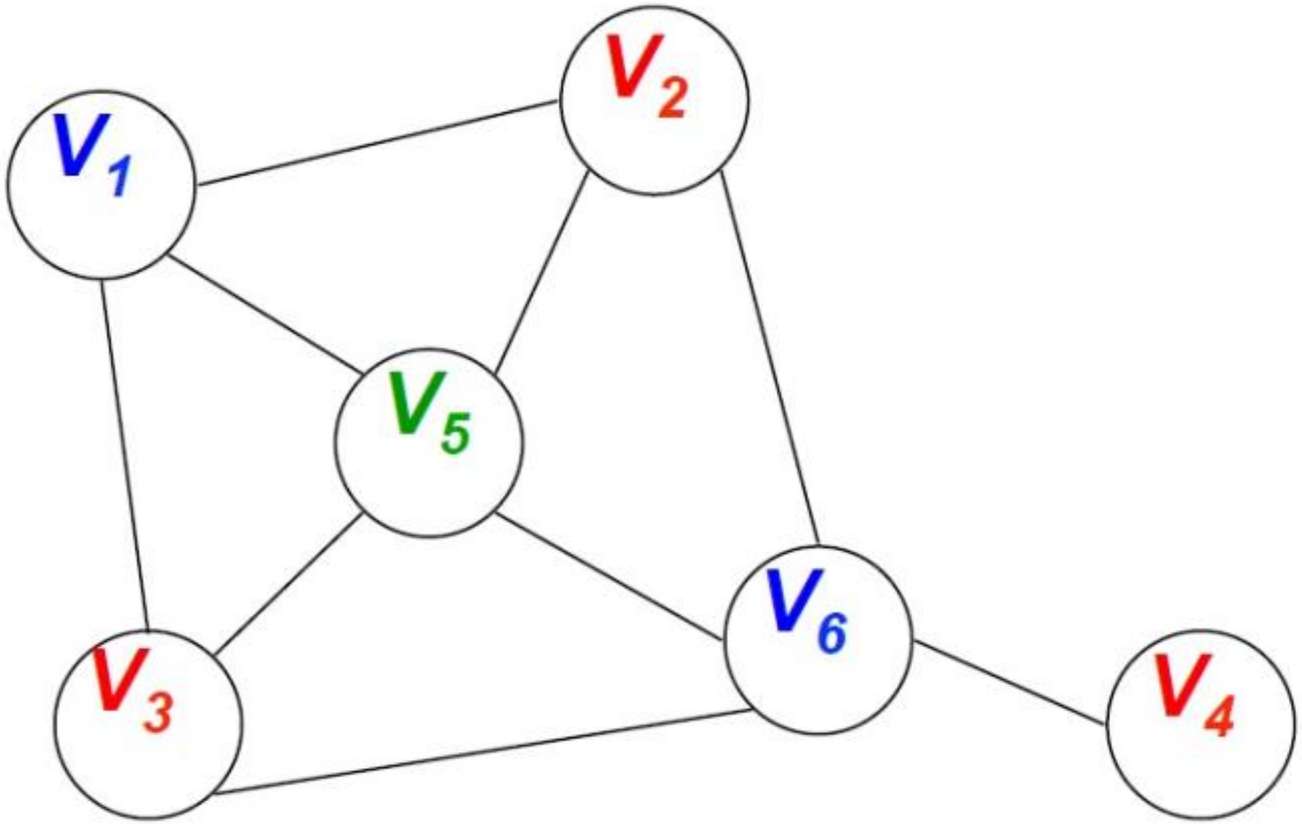


Example: Graph Coloring



- Consider N nodes in a graph
- Assign values $V_1,..,V_N$ to each of the N nodes
- The values are taken in $\{R,G,B\}$
- Constraints: If there is an edge between i and j, then V_i must be different from V_j

Example: Graph Coloring



Example: Map Coloring

- **Variables:** WA, NT, Q, NSW, V, SA, T
- **Domains:** {red, green, blue}
- **Constraints:** adjacent regions must have different colors
  e.g., WA $\neq$ NT, or (WA, NT) in {(red, green), (red, blue), (green, red), (green, blue), (blue, red), (blue, green)}

Example: Map Coloring

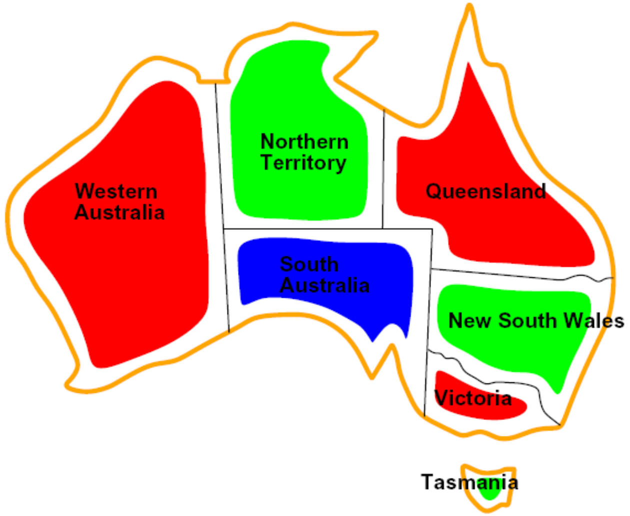
- Variables: WA, NT, Q, NSW, V, SA, T
- Domains: $D = \{\text{red}, \text{green}, \text{blue}\}$
- Constraints: adjacent regions must have different colors

  Implicit: $WA \neq NT$

  Explicit: $(WA, NT) \in \{(\text{red}, \text{green}), (\text{red}, \text{blue}), \ldots\}$

- Solutions are assignments satisfying all constraints, e.g.:

  {WA=red, NT=green, Q=red, NSW=green, V=red, SA=blue, T=green}



Uncertainty

Definition of Uncertainty

1. Lack of certainty or sureness.
2. Possible outcomes or states are not known with certainty.

Why Uncertainty Occurs

1. Incomplete information.
2. Inherent randomness in the system.
3. Ambiguity and vagueness in data or information.

Examples in Real-world Scenarios

1. Medical diagnosis.
2. Weather forecasting.
3. Financial market prediction.

Nature of Uncertainty

- General situation:
  - **Observed variables (evidence)**: Agent knows certain things about the state of the world (e.g., sensor readings or symptoms)
  - **Unobserved variables**: Agent needs to reason about other aspects (e.g. where an object is or what disease is present)
  - **Model**: Agent knows something about how the known variables relate to the unknown variables

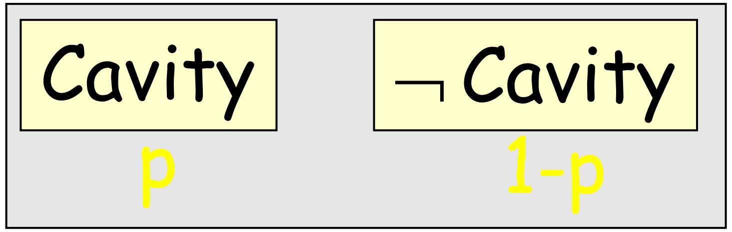
- Probabilistic reasoning gives us a framework for managing our beliefs and knowledge

Probabilistic Belief

- Consider a world where a dentist agent D meets with a new patient P
- D is only interested in whether P has a cavity; so, a state is described with a single proposition: Cavity
- Before observing P, D does not know if P has a cavity, but from years of practice, he believes Cavity with some probability p and $\neg$Cavity with probability 1-p
- The proposition is now a boolean random variable and (Cavity, p) is a **probabilistic belief**

Probabilistic Belief State

- The world has only two possible states, which are respectively described by Cavity and ¬Cavity
- The **probabilistic belief state** of an agent is a probabilistic distribution over all the states that the agent thinks possible
- In the dentist example, D’s belief state is:



Handling Uncertainty in AI

Probabilistic Models

1. Use of probability distributions to model uncertainty.
2. Bayesian Networks, Markov Models.

Fuzzy Logic

1. Deals with reasoning that is approximate rather than precise.
2. Used in natural language processing, control systems.

Possibility Theory

1. Deals with uncertainty and partial truth.
2. Used in decision making, information fusion.

Probabilistic Models

- A probabilistic model is a joint distribution over a set of random variables

- Probabilistic models:
  - (Random) variables with domains
  - Assignments are called *outcomes*
  - Joint distributions: say whether assignments (outcomes) are likely
  - *Normalized:* sum to 1.0
  - Ideally: only certain variables directly interact

Distribution over T,W

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Random Variables

- A random variable is some aspect of the world about which we (may) have uncertainty
  - R = Is it raining?
  - T = Is it hot or cold?
  - D = How long will it take to drive to work?
  - L = Where is the ghost?

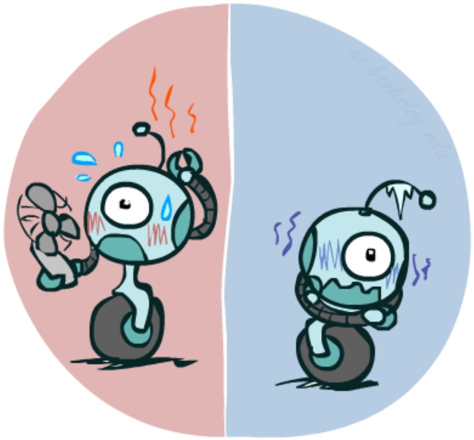
- We denote random variables with capital letters

- Like variables in a CSP, random variables have domains
  - R in {true, false}   (often write as {+r, -r})
  - T in {hot, cold}
  - D in $[0, \infty)$
  - L in possible locations, maybe {(0,0), (0,1), …}

Probability Distributions

- Associate a probability with each value

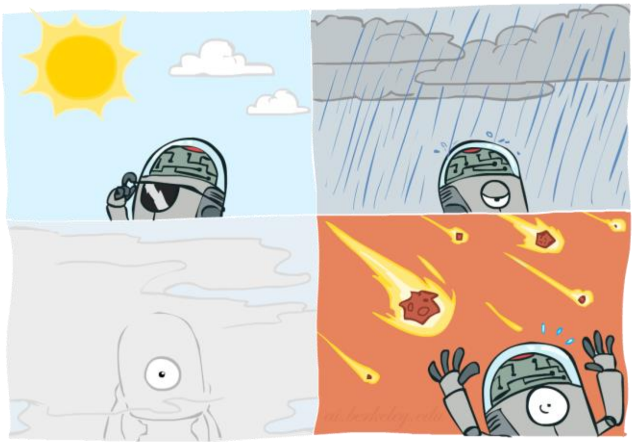
  - Temperature:



$P(T)$

T	P
hot	0.5
cold	0.5

  - Weather:



$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

~UC Berkeley

Probability Distributions

- Unobserved random variables have distributions

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

- A distribution is a TABLE of probabilities of values
- A probability (lower case value) is a single number

Shorthand notation:

$$P(hot) = P(T = hot),$$
$$P(cold) = P(T = cold),$$
$$P(rain) = P(W = rain),$$
$$\ldots$$

OK *if* all domain entries are unique

- Must have: $P(W = rain) = 0.1$

$$\forall x \;\; P(X = x) \geq 0 \qquad \sum_x P(X = x) = 1$$

Joint Distributions

- A *joint distribution* over a set of random variables: $X_1, X_2, \ldots X_n$ specifies a real number for each assignment (or *outcome*):

$$P(X_1 = x_1, X_2 = x_2, \ldots X_n = x_n)$$

$$P(x_1, x_2, \ldots x_n)$$

  - Must obey:

$$P(x_1, x_2, \ldots x_n) \geq 0$$

$$\sum_{(x_1, x_2, \ldots x_n)} P(x_1, x_2, \ldots x_n) = 1$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

Events

- An *event* is a set E of outcomes

$$P(E) = \sum_{(x_1 \ldots x_n) \in E} P(x_1 \ldots x_n)$$

- From a joint distribution, we can calculate the probability of any event
  - Probability that it's hot AND sunny?
  - Probability that it's hot?
  - Probability that it's hot OR sunny?
- Typically, the events we care about are *partial assignments*, like P(T=hot)

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3