CS 461
Artificial Intelligence

Dr. Hashim Yasin

Learning Agent

Learning Agent

- **Turing** – instead of actually programming intelligent machines by hand, build learning machines and then teach them
- All agents can improve their performance through **learning**
- Learning also allows the agent
  - to operate in initially unknown environments
  - to become more competent than its initial knowledge

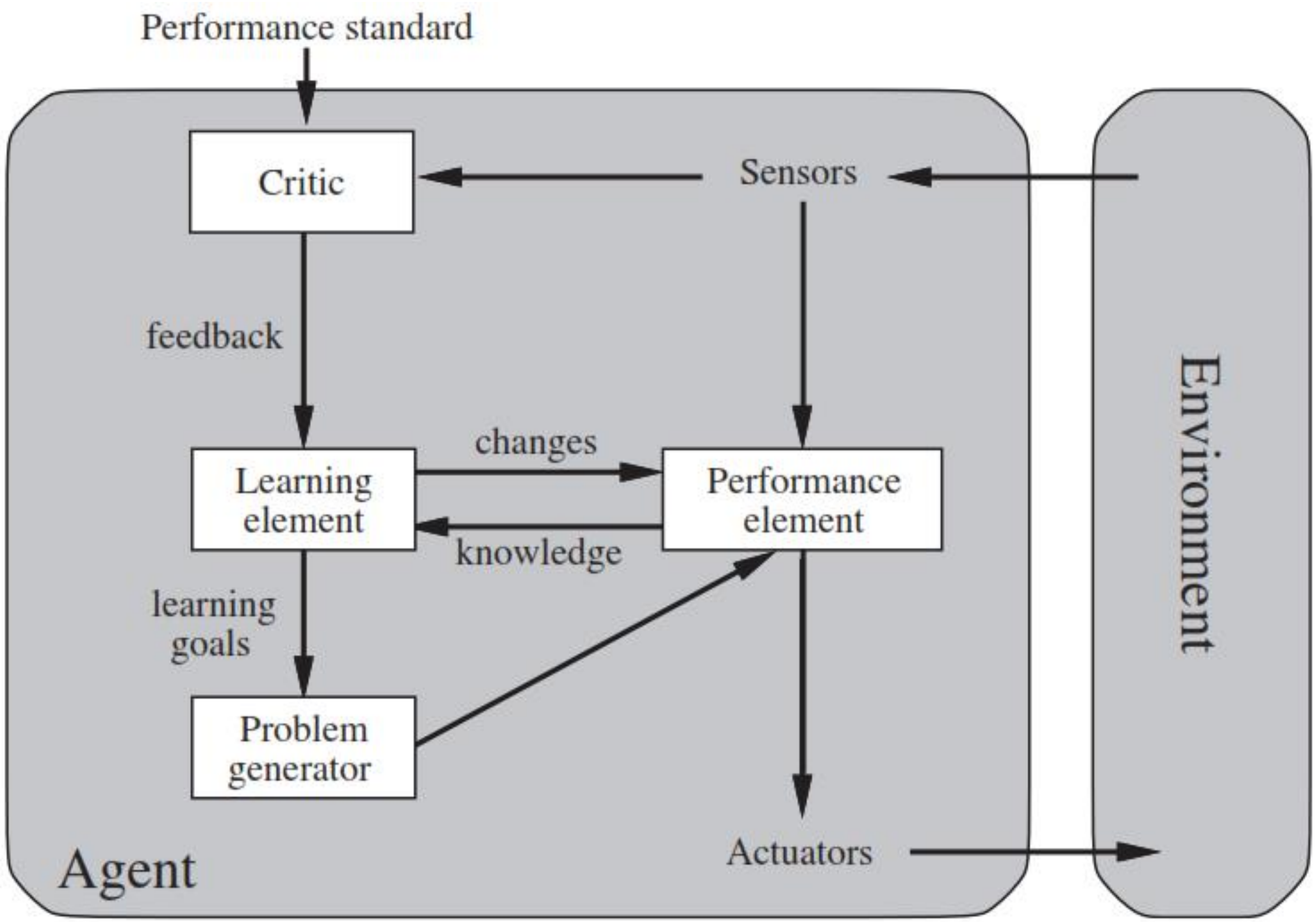
Learning Agent

- A learning agent can be divided into **four conceptual components**:
  1. The **learning element**, which is responsible for *making improvements*,
  2. The **performance element**, which is responsible for *selecting external actions*.
  3. The learning element uses feedback from the **critic**
     - how the agent is doing and
     - determines how the performance element should be modified *to do better* in the future.

Learning Agent

4. The last component of the learning agent is the **problem generator**.
    - It is responsible for suggesting actions that will lead to *new and informative experiences*.
    - The problem generator's job is to suggest these exploratory actions.

Learning Agent



State Representations

State Representations

State:

- A state is the representation of the **physical configuration**. It may be considered as all relevant aspects of the problem.
- Three ways to represent states and the transitions between them.

Atomic representation:

- A state (such as **B** or **C**) is a black box with no internal structure;
    - The algorithms underlying **search, game-playing, Hidden Markov models,** and **Markov decision processes** etc.

State Representations

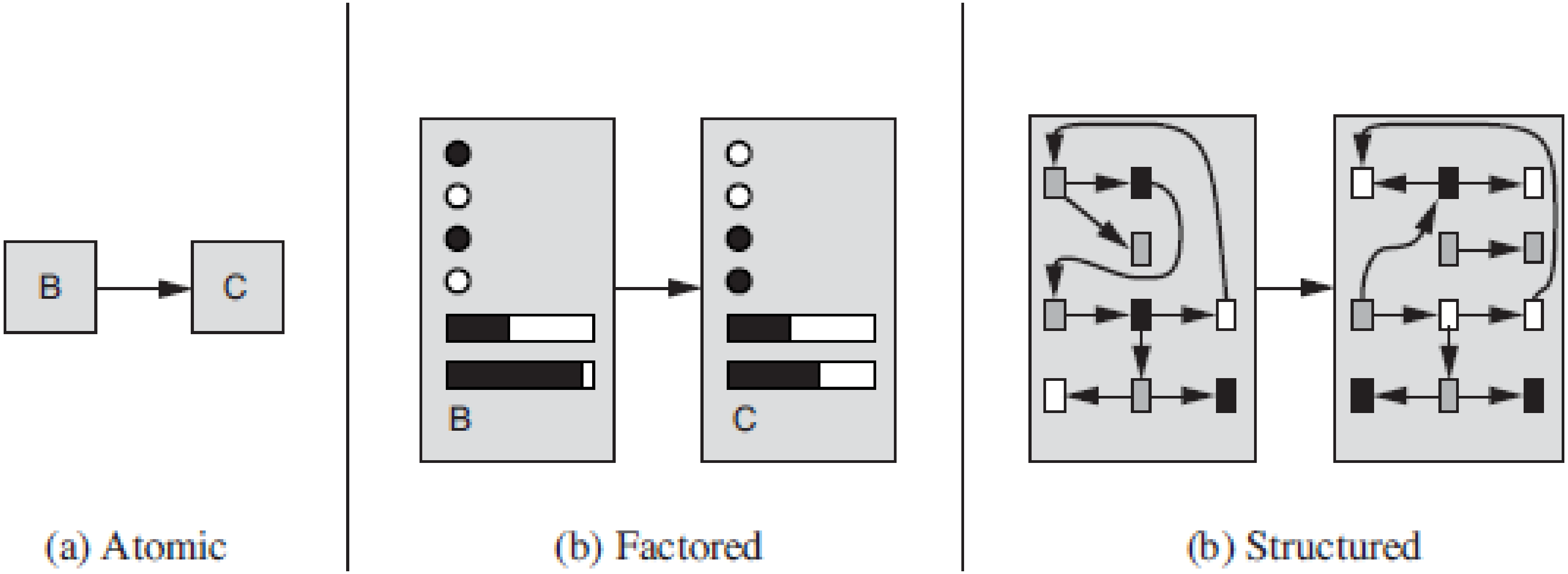
Factored representation:

- A state consists of a vector of attribute values; values can be Boolean, real valued, or one of a fixed set of symbols.
  - **Constraint Satisfaction Problems**, **Propositional Logic**, **Planning, Bayesian Networks**

Structured representation:

- A state includes objects, each of which may have attributes of its own as well as relationships to other objects
  - **First-order Logic**, **Knowledge-based Learning**

State Representations



(a) Atomic (b) Factored (b) Structured

Problem Solving Agents

Problem Solving Agents

- **Problem formulation** is the process of deciding what actions and states to consider, given a goal
- A **goal-based agent** is also called a **problem-solving agent**.
- The problem-solving agents use ***atomic representation***.
- Goal-based agents that use more advanced **factored** or **structured** representations are usually called **planning agents.**

Searching

- *How can an agent find a sequence of actions that achieves its goal when no single action will do?*

- One general approach to problem solving is
  - **to reduce the problem to be solved to one of searching a tree or graph**.

- Search plays a key role in many parts of AI.

Trees & Graphs

Tree:

- A tree is made up of **nodes and links (circles and lines)** connected so that there are NO loops (cycles).
- ***Nodes*** *are sometimes referred to as* ***vertices*** and links as **edges** (this is more common in graphs).
- A tree has a **root node** (where the tree "starts").
- Every node except the root has a single parent (aka direct **ancestor**).
- Each node except the terminal (**leaf nodes**) has one or more children (**direct descendants**).

Trees & Graphs

Graph:

- A **graph** is also a set of nodes connected by links but where **loops are allowed** and a node can have multiple parents.
- We have two kinds of graphs to deal with: Directed graphs and undirected graphs
- **Directed graphs**, where the links have direction (for example: one-way streets).
- **Undirected graphs**, where the links go both ways.

A Problem Definition

- A **problem** can be defined formally by five components:
  - Initial State
  - Actions
  - Transition Model
  - Goal Test
  - Path Cost

A Problem Definition

- **State:** A state is the representation of the **physical configuration**. It may be considered as all relevant aspects of the problem.
- **Actions:** Actions are operators:
  - **Actions are deterministic,** when we know exactly the state after an action is performed.
  - **Actions are discrete**, when we don't have to represent what happens while the action is happening.
    - For example, we assume that a flight gets us to the scheduled destination and that what happens during the flight does not matter.

A Problem Definition

- **Goal Test:** It determines whether a given state is a goal state.
- In general, we need a test for the goal, *not just one specific goal state*.
- For example,
  - We might be interested in any city in Germany rather than specifically Frankfurt.
  - When proving a theorem, all we care is about knowing one fact in our current database of facts.
  - Any final set of facts that contains the desired fact is a proof.
- **Path Cost:** Some penalty has been allocated.

A Problem Definition

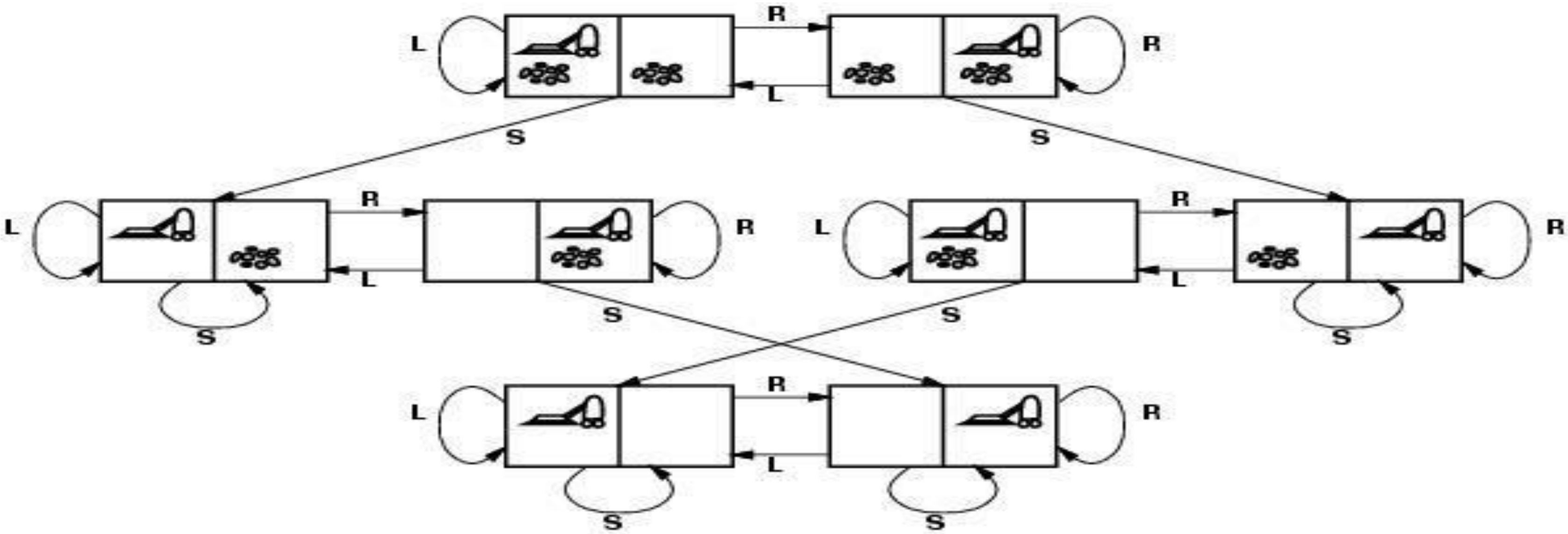
Transition Model:

- A description of what does each action do;
- The **transition model**, specified by a function $RESULT(s, a)$ that returns the state that results from doing action a in state s.
- We also use the term **successor** to refer to any state reachable from a given state by a single action

$$RESULT(In(FSD), Go(LHR)) \;=\; In(LHR)$$

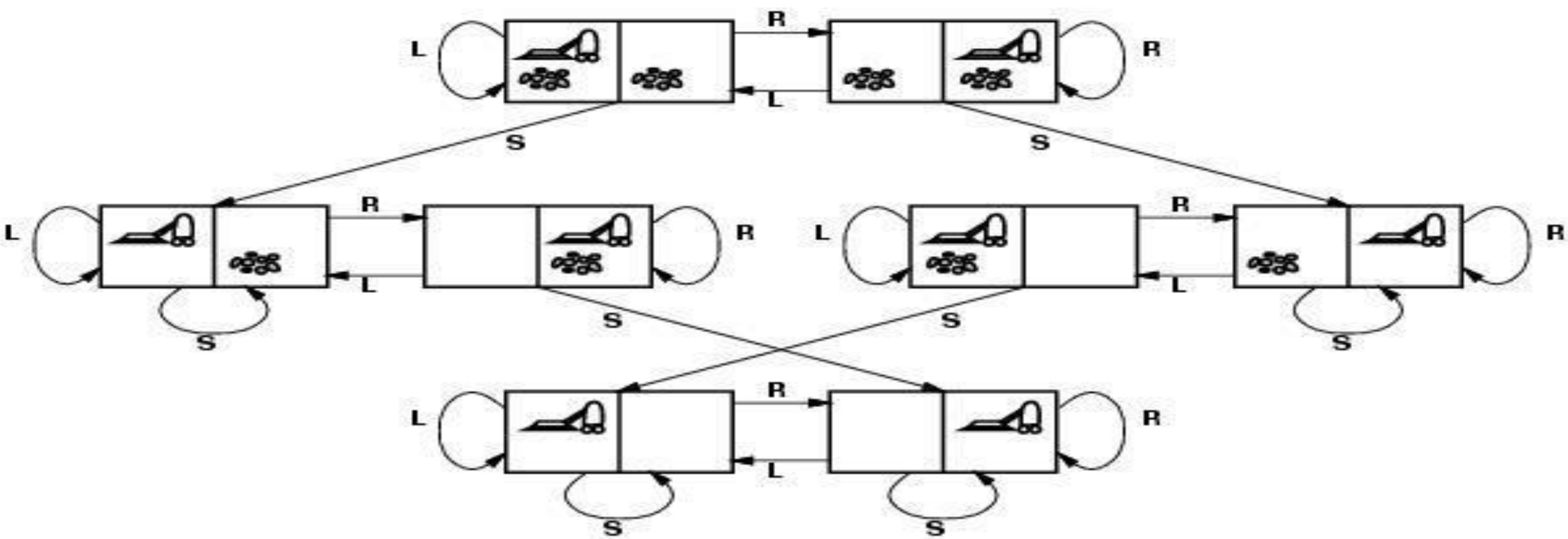
Toy Problems

Example: vacuum world



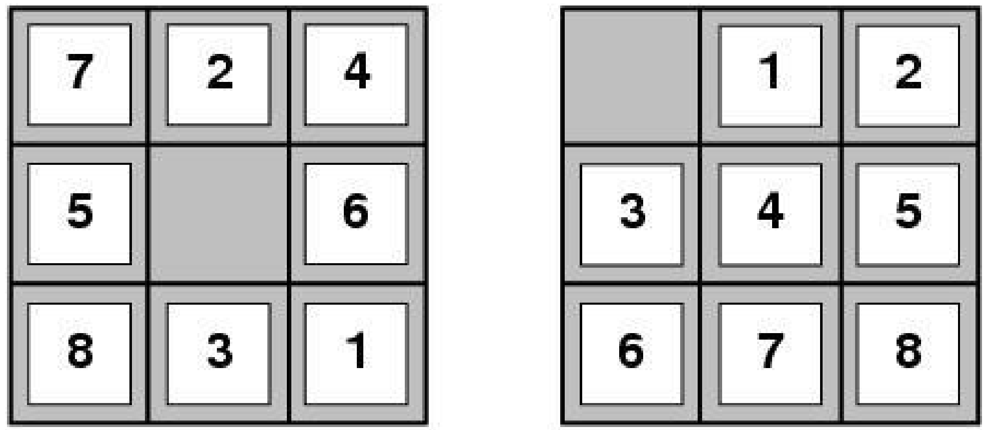
- **States:** The state is determined by both the agent location (2) and the dirt locations (2^2): two locations with or without dirt: **2×2^2=8 states**.
  - larger environment with **n** locations has **$n \times 2^n$ states**
- **Initial state:** Any state can be initial

Example: vacuum world



- **Actions:** {*Left, Right, Suck*}
- **Transition model**: The actions have their expected effects, except that moving *Left*, moving *Right*, and *Suck*ing in a clean square have no effect.
- **Goal test:** Check whether all squares are clean.
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

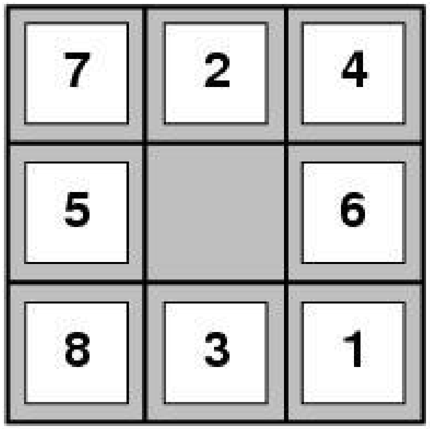
Example: 8-puzzle Problem



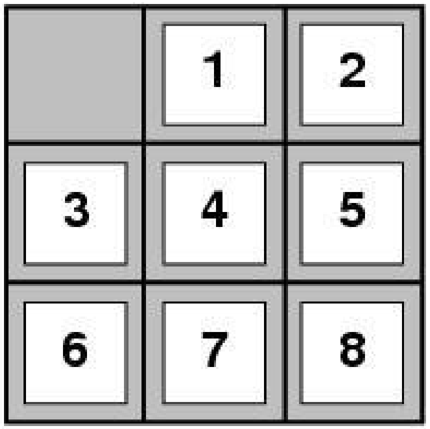
Start State Goal State

- **States:** Integer location of each tile
  - A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state:** Any state can be initial
- **Actions:** {*Left, Right, Up, Down*}

Example: 8-puzzle Problem



Start State



Goal State

- **Transition model**: Given a state and action, this returns the resulting state;
  - for example, if we apply *Left* to the start state in Figure, the resulting state has the 5 and the blank switched.
- **Goal test:** Check whether goal configuration is reached.
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

Example: 8-puzzle Problem

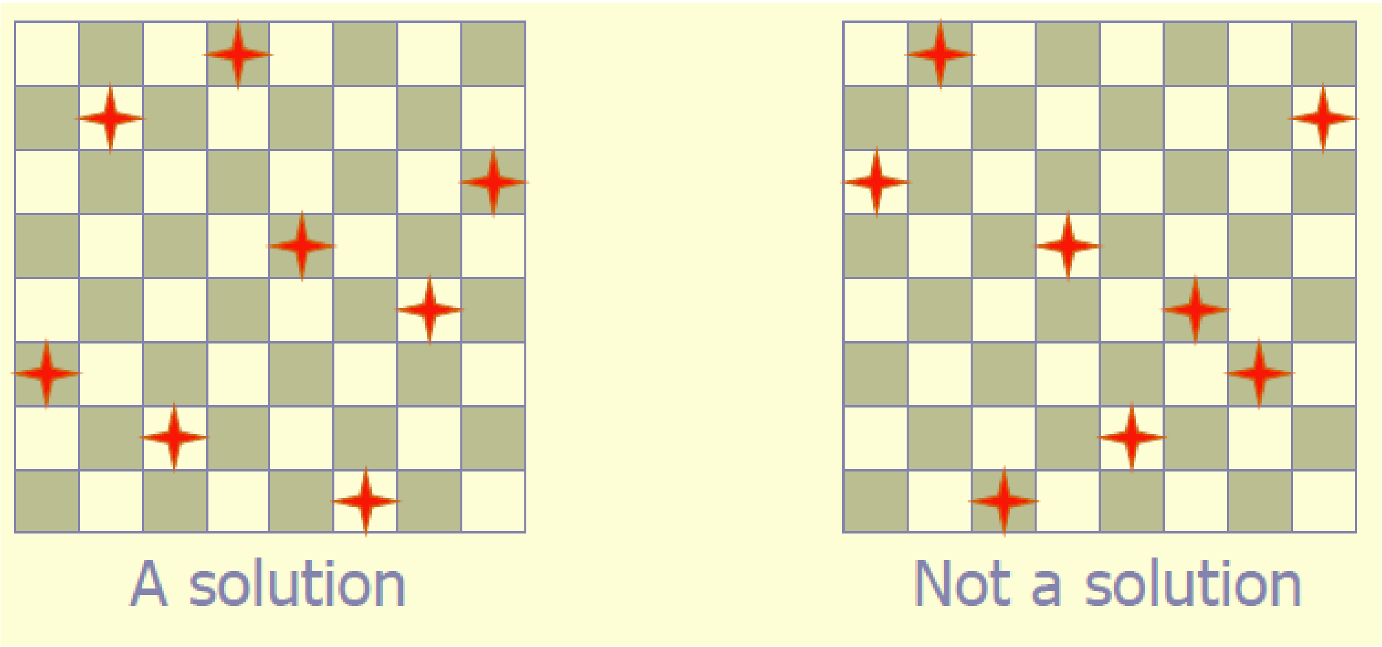
- The **8-puzzle** has 181, 440 reachable states and is easily solved.

- The **15-puzzle** (on a 4×4 board) has around 1.3 trillion states, and random instances can be solved optimally in a few milliseconds by the best search algorithms.

- The **24-puzzle** (on a 5 × 5 board) has around 10^{25} states, and random instances take several hours to solve optimally.

Example: 8-queen Problem

- Place 8 queens in a chessboard so that no two queens are in the same row, column, or diagonal.



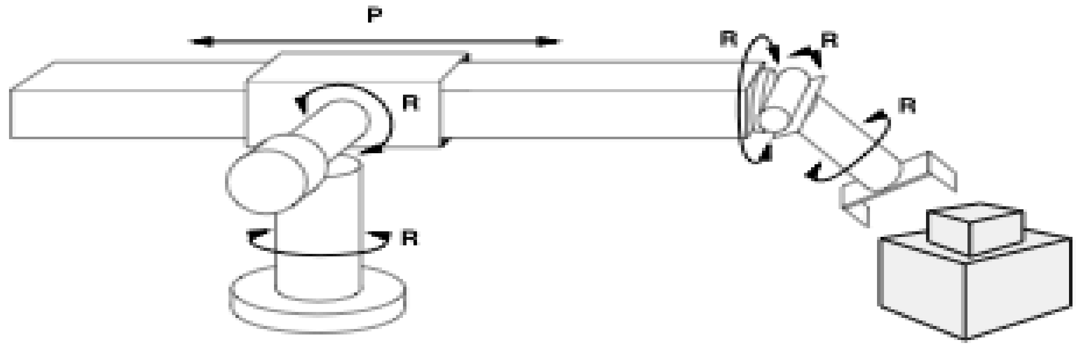
Example: 8-queen Problem

- **States:** any arrangement of 0 to 8 queens on the board
- **Initial state:** No queen on the board
- **Transition model**: Returns the board with a queen added to the specified square.
- **Actions:** add a queen in any square
- **Goal test:** 8 queens on the board, none attacked
- **Path cost:** none

64^8 states with 8 queens

Real Life Problems

Example: Robot Assembly



- **States:** Real-valued coordinates of robot joint angles; parts of the object to be assembled.
- **Initial state:** Any arm position and object configuration.
- **Actions:** Continuous motion of robot joints
- **Goal test:** Complete assembly
- **Path cost:** Time to execute

Example: Route Finding Problem

- **States:** each state is represented by **a location** (e.g., An airport) and the **current time**

- **Initial state:** specified by the problem

- **Actions**: *Take any flight from the current location*, in any seat class, leaving after the current time, leaving enough time for within-airport transfer if needed.

Example: Route Finding Problem

- **Transition model**: The state resulting from taking a flight will have the *flight's destination* as the current location and the *flight's arrival time* as the current time.
- **Goal test:** are we at the destination by some pre-specified time
- **Path cost:** monetary cost, waiting time, flight time, customs and immigration procedures, seat quality, time of day, type of airplane, frequent-flyer mileage awards, etc.

Searching

Uninformed search

- No information about the number of steps
- No path cost taken into account from current state to the goal
- Search the state space **blindly**

Informed search, or heuristic search

- A cleverer strategy is used that searches toward the goal.
- Based on the information from the current state.

Searching

Any-path searches and optimal searches.

- **Any-path searches** will just settle for finding some solution,

- **Optimal searches** are looking for the best possible path.

Evaluation of a Search Strategy

- **Completeness:** is the strategy guaranteed to find a solution when there is one?
- **Optimality:** does the strategy find the highest-quality (least-cost) solution when there are several different solutions?
- **Time complexity:** how long does it take to find a solution?
- **Space complexity:** how much memory is needed to perform the search?

Evaluation of a Search Strategy

Complexity may be expressed in

- **b,** branching factor, maximum number of successors of any node
- **d**, the depth of the shallowest goal node. **(number of steps along the path from the root),** depth of the least-cost solution
- **m**, the maximum length of any path in the state space

Time and Space is measured in

- number of nodes generated during the search
- maximum number of nodes stored in memory

Reading Material

- **Artificial Intelligence,** A Modern Approach

  Stuart J. Russell and Peter Norvig
  - **Chapter 2 & 3.**