Problem Solving by Search in AI

Md Rifat Ahmmad Rashid, Associate Professor, EWU

1 Introduction to Search in AI

In Artificial Intelligence, **search** refers to the process of systematically exploring a **state space** to find a sequence of actions that leads from a starting state to a goal state (Fundamentals of Artificial Intelligence) (Fundamentals of Artificial Intelligence).

A **search problem** is typically defined by five components:

- an **initial state** (where the agent starts),
- a set of **actions** available in each state,
- a **transition model** (describing the result of each action),
- a **goal test** (to determine if a given state is a goal),
- a **path cost** function (assigning a numeric cost to each path) (Fundamentals of Artificial Intelligence)

Formally, the ***state space*** 1 is the set of all states reachable from the initial state by any sequence of actions, which can be conceptualized as a graph where nodes are states and edges represent actions leading to successor states (Fundamentals of Artificial Intelligence).

- A **solution** to a search problem is an *action sequence* that transforms the initial state into a goal state (Fundamentals of Artificial Intelligence).
- Among all solutions, an **optimal solution** is one with minimum path cost (Fundamentals of Artificial Intelligence).

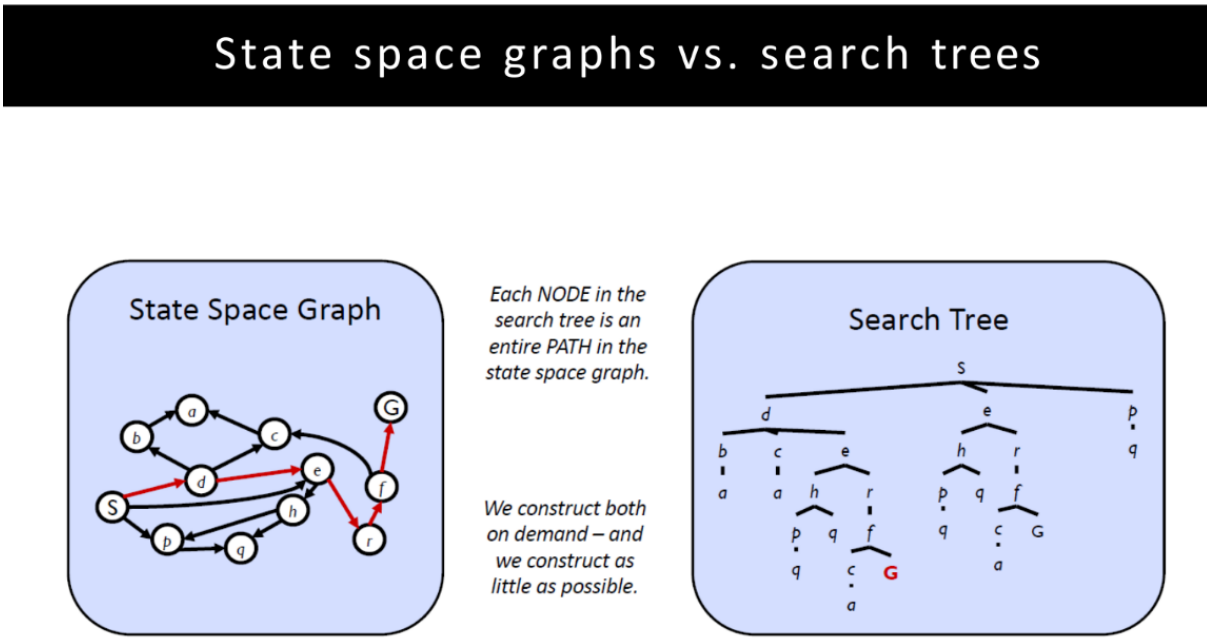

Figure 1: State Space graphs vs Search Tree

1.1 Example

For example, the classic **8-puzzle** problem 2 consists of a 3×3 board with eight numbered tiles and one empty space; the initial state can be any scrambled configuration of tiles, and the goal is to reach a configuration (such as the tiles in numerical order) by sliding tiles into the empty space (BFS, DFS, Uniform Cost Search and A* Algorithms: Mastering Search Techniques in AI Problem Solving — CamelEdge) (BFS, DFS, Uniform Cost Search and A* Algorithms: Mastering Search Techniques in AI Problem Solving — CamelEdge).

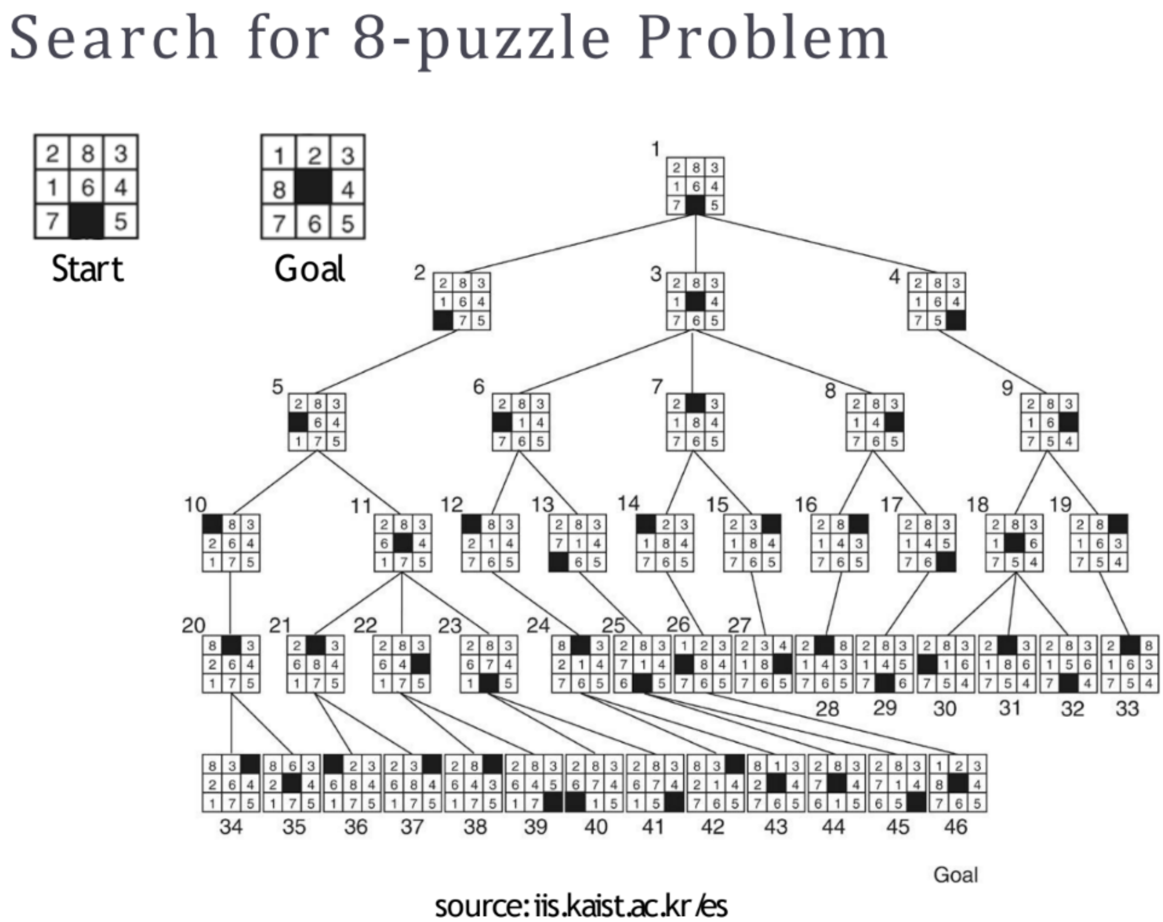

Figure 2: 8-puzzle problem: initial and goal state

Another example is **pathfinding** on a map: states represent locations, actions are movements (e.g. driving along roads), and the goal is to navigate from a start location to a destination. Other examples include **scheduling problems** (where states are partial schedules and actions add scheduling decisions) and **robot navigation** tasks (where a robot must plan movements to reach a target location).

In all these cases, the agent must *search* through possible sequences of actions to find a successful (and possibly optimal) path to the goal.

2 Search algorithms

Search algorithms explore the state space by expanding states (applying actions to generate successors) and systematically considering different action sequences.

- **Search tree:** The set of all possible action sequences forms a **search tree** (with the initial state as the root) (Search). B**ecause real-world state** spaces are often enormous, **a key challenge is designing efficient search strategies that find solutions in a reasonable time.**

Basic Idea

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

Algorithm 1 Tree Search Algorithm

```
function TREE-SEARCH(problem)
    Initialize the frontier using the initial state of problem
    while true do
        if frontier is empty then
            return failure
        end if
        Choose a leaf node and remove it from the frontier
        if the node contains a goal state then
            return the corresponding solution
        end if
        Expand the chosen node, adding the resulting nodes to the frontier
    end while
end function
```

Frontier: all leaf nodes available for expansion at any given point.

Different data structures (e.g., FIFO, LIFO) for frontier can cause different orders of node expansion and thus produce different search algorithms.