PROBLEM SOLVING & SEARCH STRATEGY
Part 1

Dr. Emad Natsheh

Problem solving

- We want:
  - To automatically solve a problem
- We need:
  - A representation of the problem
  - Algorithms that use some strategy to solve the problem defined in that representation

Problem Description

- Components
  - ✓ State space
    - ✓ Initial state
    - ✓ Goal state
  - ✓ Actions (operators)
  - ✓ Path cost

States

- A problem is defined **by its elements and their relations**
- A state is a representation of those elements in a given moment.
- Two special states are defined:
  - **Initial state** (starting point)
  - **Goal state**

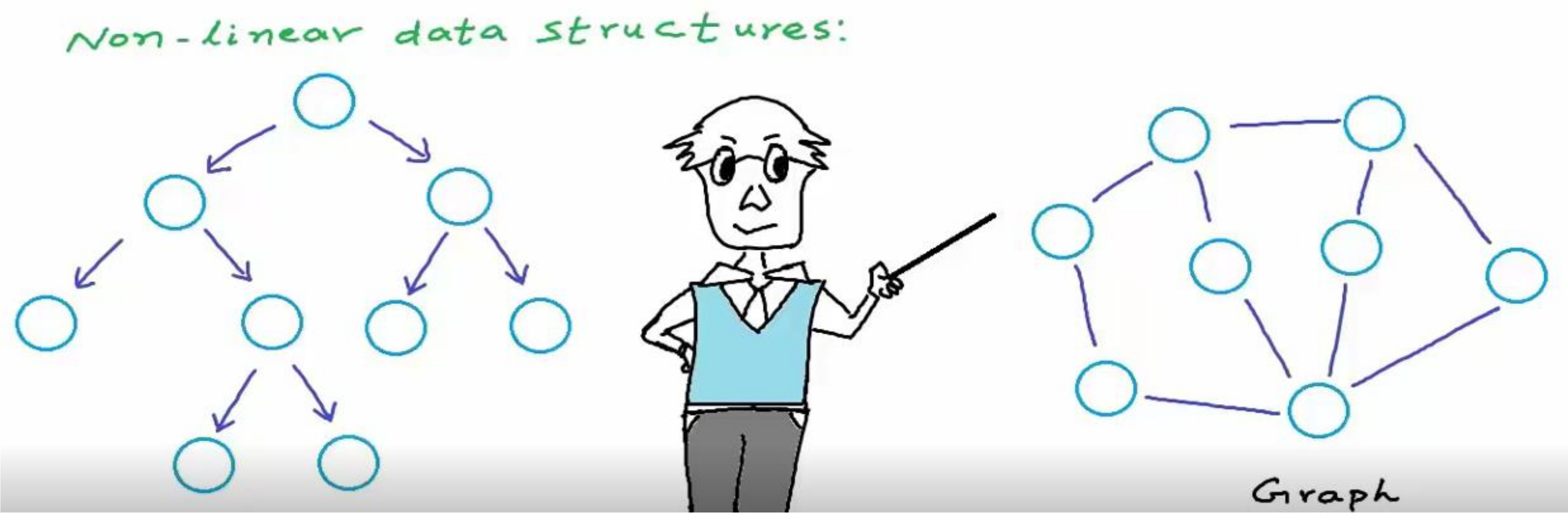
State Modification: Successor Function

- A successor function is needed to move between different states.
- A successor function is a description of possible actions a set of operators. It is a transformation function on a state representation, which move it into another state.
- The successor function defines a relation of accessibility among states.

State space

- The state space is the set of all states reachable from the initial state
- Its form a graph (or tree) in which the nodes are states and the arcs between nodes are actions.
- A path in the state space is a sequence of states connected by a sequence of actions.
- The solution of the problem is part of the map formed by the state space.

Tree vs Graph



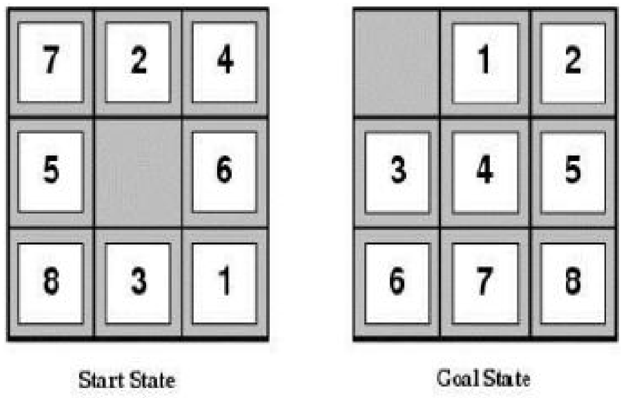
if N nodes
then (N-1) edges
one edge for each
parent-child relationship

Problem Solution

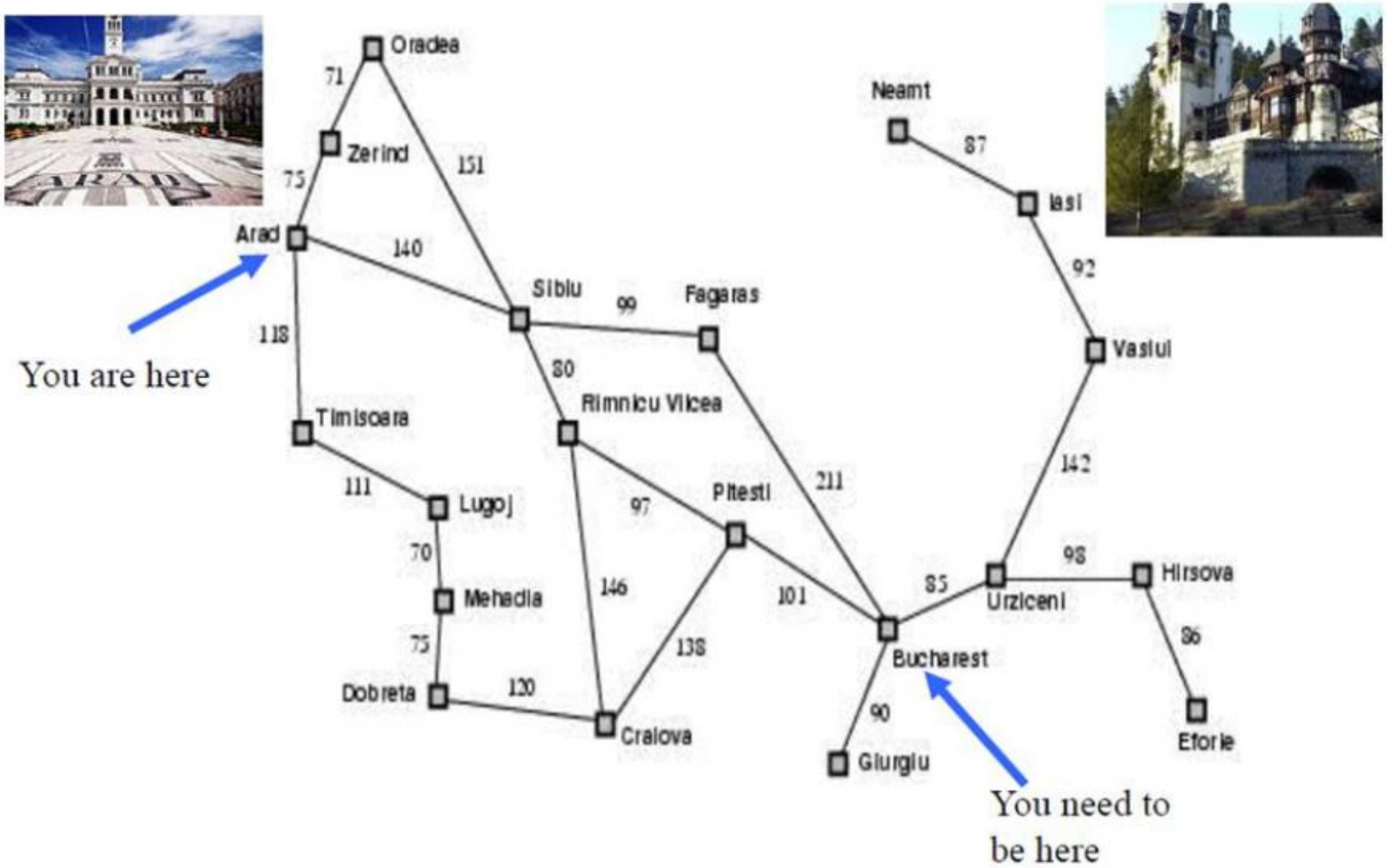
- A solution in the state space is a path from the initial state to a goal state
- Path/solution cost: function that assigns a numeric cost to each path, the cost of applying the operators to the states
- Solution quality is measured by the path cost function, and an optimal solution has the lowest path cost among all solutions.

Example 8-puzzle

- **State space**: configuration of the eight tiles on the board
- **Initial state *as shown***
- **Goal state**: as shown
- **Operators or actions**: “blank moves”
  - Condition: the move is within the board
  - Transformation: blank moves *Left*, *Right*, *Up*, or *Down*
  - Performance measure: minimize total moves
- **Find solution**: Sequence of pieces moved: 3,1,6,3,1,…
  - optimal sequence of operators



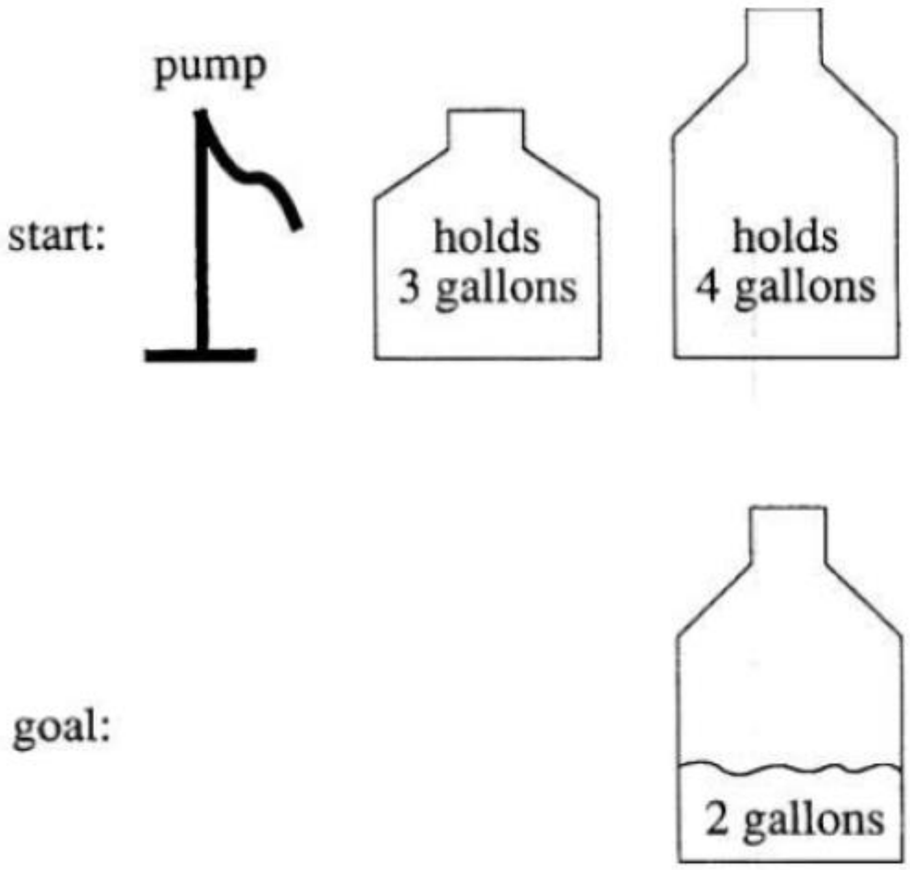
Example: Travelling



Problem Representation

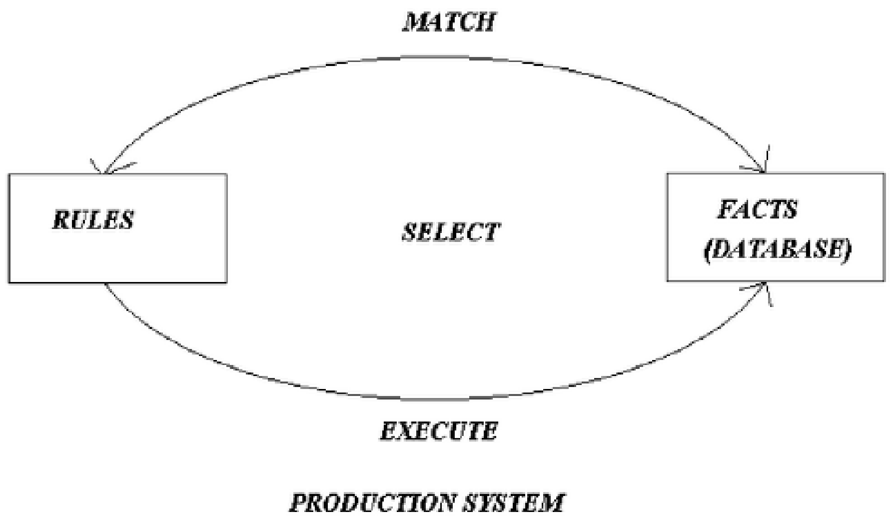
Example Water Jug Problem

- You have a 4-gallon and a 3-gallon water jug
- You have a faucet with an unlimited amount of water
- You need to get exactly 2 gallons in 4-gallon jug



Problem Description

- State representation: **(x, y)**
  - x: Contents of four gallon
  - y: Contents of three gallon
- Start state: **(0, 0)**
- Goal state **(2, n)**
- Operators
  - Fill 3-gallon from faucet, fill 4-gallon from faucet
  - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
  - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
  - Dump 3-gallon down drain, dump 4-gallon down drain

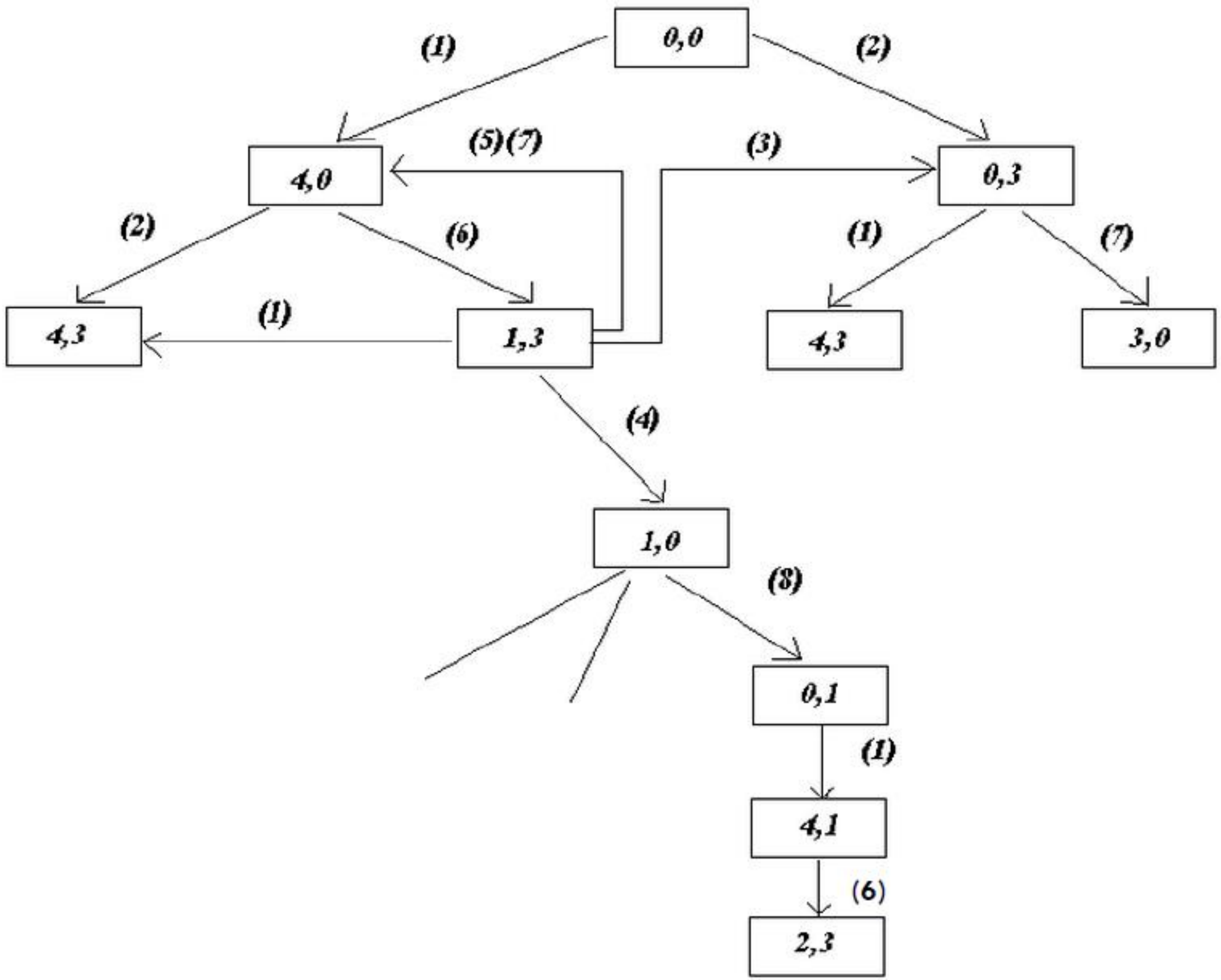


Operations (Actions)

#	Actions
1	Fill X from Pump
2	Fill Y from Pump
3	Empty X into Ground
4	Empty Y into Ground
5	Get water from Y into X until X is full
6	Get water from X into Y until Y is full
7	Get all water from Y into X
8	Get all water from X into Y

Rules

#	Rules
1	
2	
3	
4	
5	
6	
7	
8	


0,0
(1)
(2)
4,0
(5)(7)
(3)
0,3
(2)
(6)
(1)
(7)
4,3
(1)
1,3
4,3
3,0
(4)
1,0
(8)
0,1
(1)
4,1
(6)
2,3

Another Solution to the Water Jug Problem

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	

Algorithm for Problem Solving

1. **Initialize the search tree using the initial state of the problem**
2. **Choose a terminal node for expansion according to certain search strategy**
   - ❑ **If no terminal node is available for expansion return failure**
   - ❑ **If the chosen node contains a goal return the node**
3. **Expand the chosen node (according to the rules) and add the resulting node to the search tree**
4. **Go to step 2**

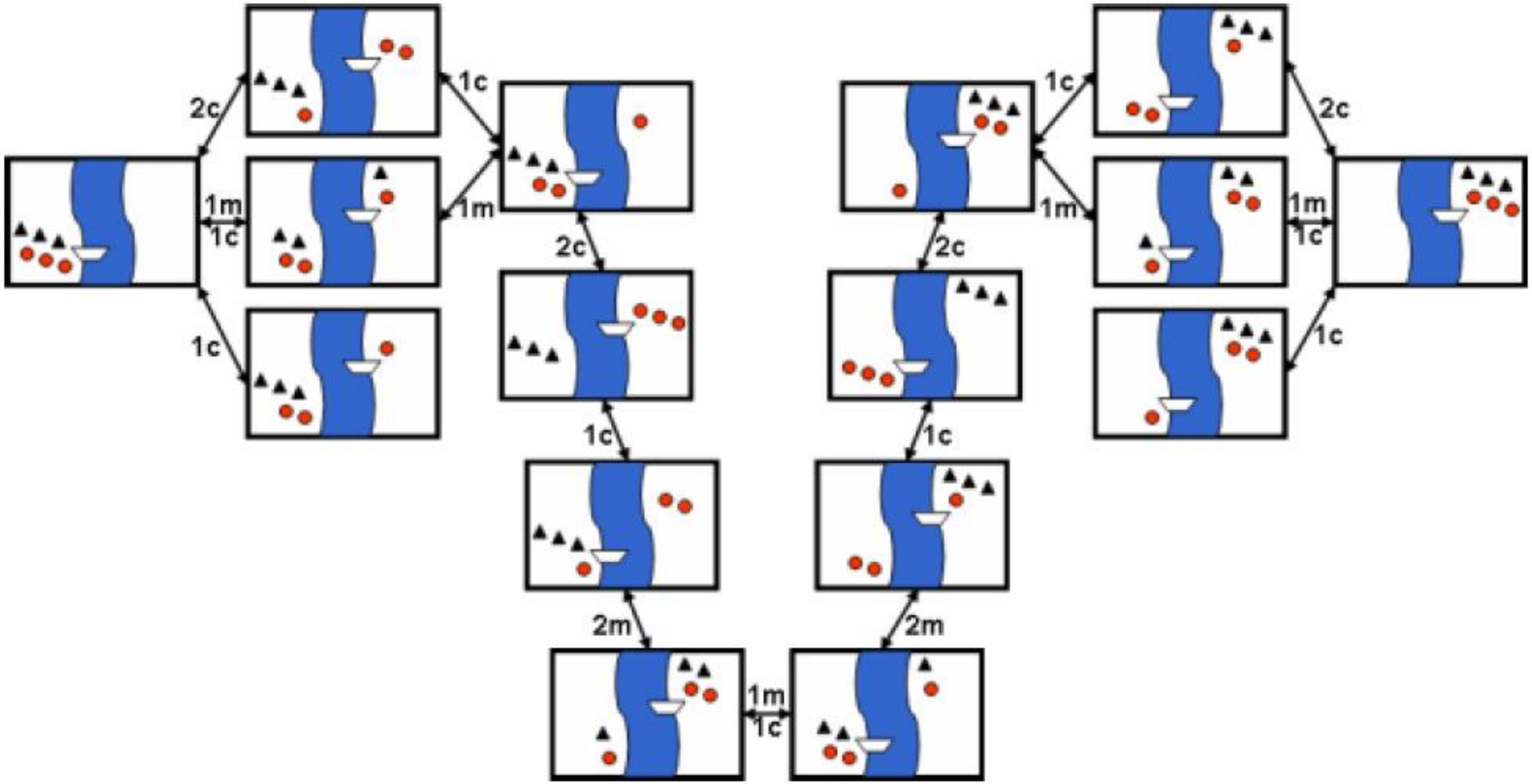
Missionaries & Cannibals Problem

Problem Description

- **State(# of missionaries Left, # of cannibals Left, # of missionaries Right, # of cannibals Right, side_of_the_boat)**
- **Initial State => State (3, 3, 0, 0, 0)**
- **Final State => State (0, 0, 3, 3, 1).**
- **Actions**
  - Carry (2, 0).
  - Carry (1, 0).
  - Carry (1, 1).
  - Carry (0, 1).
  - Carry (0, 2).
    Where Carry (M, C) means the boat will carry M missionaries and C cannibals on one trip.

Rules

#	Rules
1	One missionaries can move only when __________ in one side And __________ in the other
2	Two missionaries can move only when _________in one side And __________ in the other
3	One cannibals can move only when ____________ in one side And __________ in the other
4	Two cannibals can move only when ____________in one side And _________ in the other
5	One missionary and one cannibal can move only when _________ in one side And _________ in the other



2c
1c
1m
1c
1m
1c
1m
2c
1c
2m
1m
1c
2m
1c
2c
1c
1m
2c
1m
1c
1c

Vacuum Cleaner

- **World state space**
- **State**
- **Actions**
- **Goal**
- **Path costs:**