SUBJECT : COMPUTER APPLICATIONS

CLASS : MCA YEAR/SEMESTER : SIXTH SEMESTER

NAME OF PAPER : ARTIFICIAL INTELLIGENCE

TOPIC : **SEARCH TECHNIQUES**

SUB-TOPIC : **PROBLEMS,PROBLEM SPACES AND SEARCH**

KEYWORDS : **STATE SPACE , GOAL STATE,OPERATORS , WATER-JUG PROBLEM , CONTROL STRATEGY**

CREATED BY :

Dr.(Mrs.) Satya Singh

Professor /Head, Department Of Computer Science & Applications &

Dean, Faculty Of Science & Technology,

M.G.Kashi Vidyapith , Varanasi

satyasinghmgkvp@gmail.com

Problems , Problem Spaces and Search

Defining a Search Problem

- State space S: all possible configurations of the domain of interest
  - An initial (start) state $s0 \in S$
  - Goal states $G \subset S$: the set of end states
    - Often defined by a goal test rather than enumerating a set of states
  - Operators A: the actions available
    - Often defined in terms of a mapping from a state to its successor

Formalizing Search in a State Space

- Everything in AI comes down to search.
- Goal: understand search, and understand how.
  - A *state space is a*

    graph (V, E):

    V is a set of nodes

    E is a set of arcs

    Each arc is directed

    from a node to another node

Formalizing Search in a State Space

V: A node is a data structure that contains a state description plus other information such as the parent of the node, the name of the operator that generated the node from that parent, and other bookkeeping data

E: Each arc corresponds to an instance of one of the operators. When the operator is applied to the state associated with the arc's source node, then the resulting state is the state associated with the arc's destination node

Formalizing Search

- Each arc has a fixed, positive cost
  Corresponding to the cost of the operator
  What is “cost” of doing that action?
- Each node has a set of successor nodes
  Corresponding to all operators (actions) that can apply at source node’s state
- Expanding a node is generating successor nodes, and adding them (and associated arcs) to the state-space graph

Formalizing Search

- One or more nodes are designated as start nodes

- A goal test predicate is applied to a *state to*

determine if its associated node is a goal node

Formal description of a problem

Define a state space that contains all possible configurations of the relevant objects, without enumerating all the states in it. A *state space* represents a problem in terms of *states* and *operators* that change states

Define some of these states as possible initial states;

Specify one or more as acceptable solutions, these are goal states;

Specify a set of rules as the possible actions allowed. Each rule results in a specific state to be expanded into its successor node, which is now included in the state space graph.

Problem Size

- The size of a problem is specified in terms of the number of possible states . For example :

  Tic-Tac-Toe has about 3^9 states.

  Checkers has about 10^{40} states.

  Rubik's Cube has about 10^{19} states.

  Chess has about 10^{120} states in a typical game.

Control Strategy

- There is a **control strategy** which is a structure used by the AI program to facilitate the

  search by guiding it in the right direction and reducing search time for reaching the goal state.

Requirements of a Control Strategy

❖ It should cause motion(Transition between states).

❖ It should be systematic(Cycles should not occur in the search space as they form loops , preventing effective motion towards the Goal).

State-space Search

- State-space search is the process of searching through a state space for a solution by making explicit a sufficient portion of an implicit state space graph to find a goal node
- Initially V={S}, where S is the start node
- When S is expanded, its successors are generated; those nodes are added to V and the arcs are added to E
- This process continues until a goal node is found
- It is not practical to represent the entire space

An Example :The water jug problem

- There are two jugs called **four** and **three** ; four holds a maximum of four gallons and **three** a maximum of three gallons. How can we get 2 gallons in the jug **four**?
- The state space is a set of ordered pairs giving the number of gallons in the pair of jugs at any time ie (**four,three**) where **fou**r = 0, 1, 2, 3, 4 and **three** = 0, 1, 2, 3.
- The start state is (0,0) and the goal state is

  (2,n) where n is a don't care but is limited to **three** holding from 0 to 3 gallons.

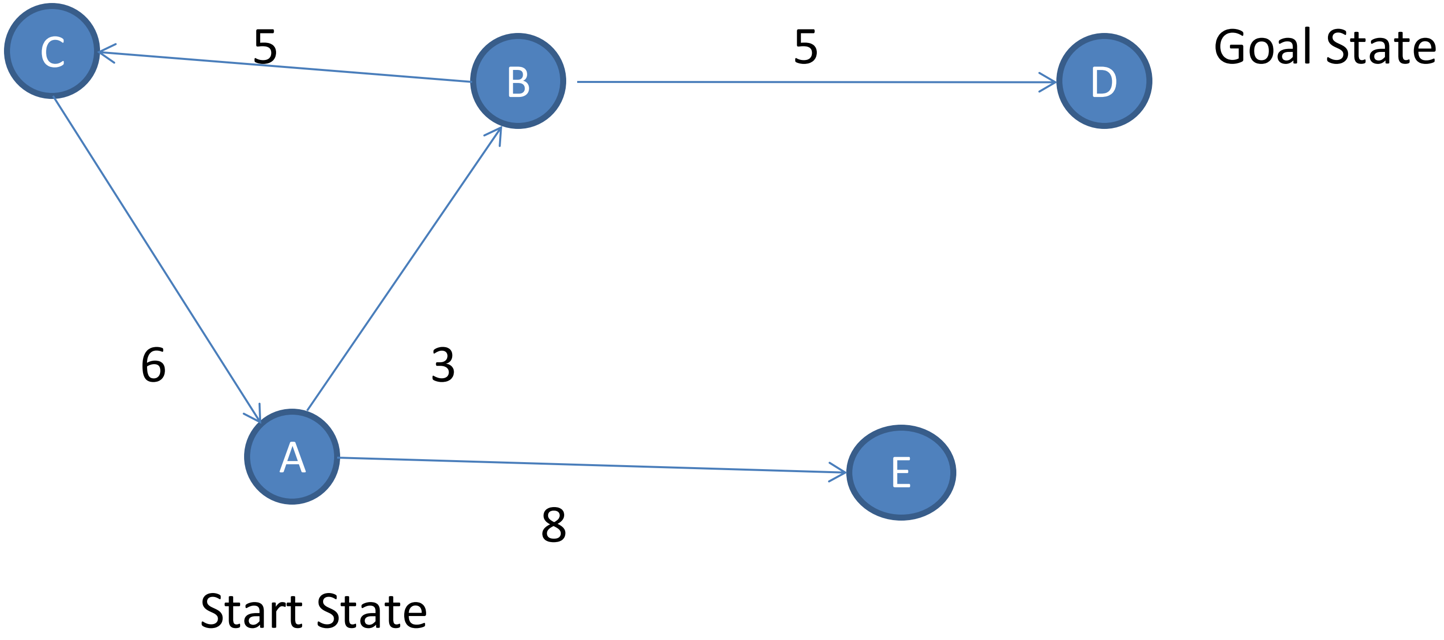
	Initial	condition	goal	comment
1.	(four,three)	if four < 4	(4,three)	fill four from tap
2.	(four,three)	if three< 3	(four,3)	fill three from tap
3.	(four,three)	If four > 0	(0,three)	empty four into drain
4.	(four,three)	if three > 0	(four,0) empty three into drain	if three > 0 (four,0) empty three into drain
5.	(four,three)	if four+three<4	(four+three,0)	empty three into four
6.	(four,three)	if four+three<3	(0,four+three)	empty four into three
7.	(0,three)	If three>0	(three,0)	empty three into four
8.	(four,0)	if four>0	(0,four)	empty four into three
9.	(0,2)		(2,0)	empty three into four
10.	(2,0)		(0,2)	empty four into three
11.	(four,three)	if four<4	(4,three-diff)	pour diff, 4-four, into four from three
12.	(four,three)	if three<3	(four-diff,3)	pour diff, 3-three, into three from four

A Solution

Jug four	Jug Three	Rule Applied
0	0	
0	3	2
3	0	7
3	3	2
4	2	11
0	2	3
2	0	10

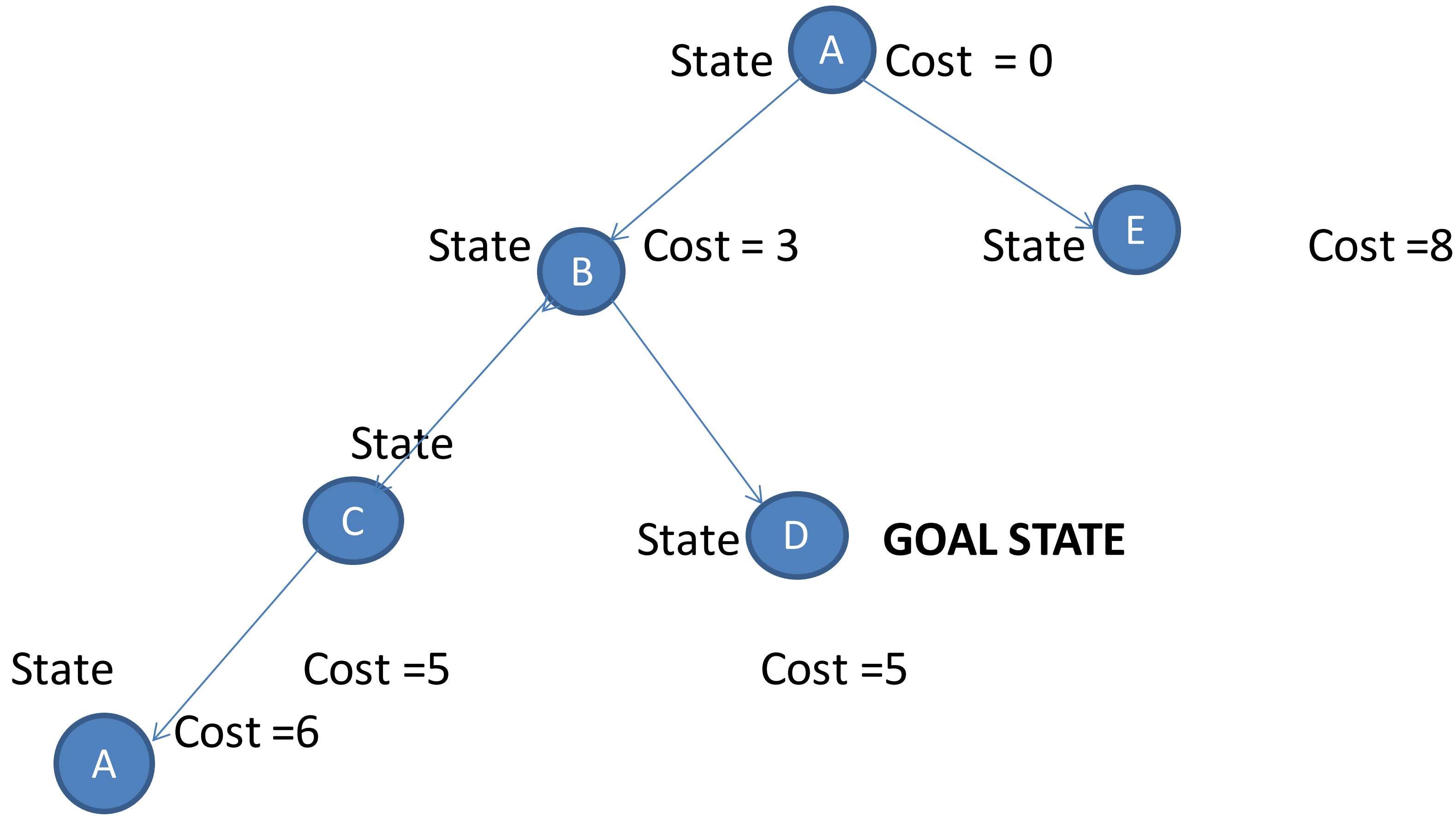
GRAPHS AND THEIR SEARCH TREES

Let us consider the following graph :



GRAPHS AND THEIR SEARCH TREES

The search tree of the graph :



TYPES OF SEARCH

- Uninformed (Blind)Search

- Informed(Guided) Search   or

  Heuristic Search

Uninformed (blind) search

- If a state is not a goal, we cannot estimate its closeness to the goal
- Hence, all we can do is move systematically between states until we reach a goal state
- In contrast, informed (heuristic) search uses a guess on how close to the goal a state might be

Uninformed search methods

- Breadth-First Search
- Depth-First Search
- Iterative Deepening

Implementation Details

We need to keep track only of the nodes that need to be expanded - known as the frontier or open list

This can be implemented using a (prioritized) queue:

1. Initialize the queue by inserting the node for the initial state
2. Repeat

(a) If the queue is empty, return failure

(b) Dequeue a node

(c) If the node contains a goal state, return the path

(d) Otherwise expand the node, inserting the resulting nodes into queue

We can observe that various Search algorithms differ in their queuing function

Characteristics of Search Strategies

- Completeness:

  If a solution exists , the search strategy guarantees to find it

- Time complexity:

  The time taken to find a solution in the average and worst case . Typically , it is measured in number of states visited/nodes expanded

- Space complexity:

  How much space is used by the algorithm?

  Typically , it is measured in maximum size of the "nodes" list during search

- Optimality/Admissibility

  If a solution is found, is it guaranteed to be optimal (the solution with minimum cost)?

REFERENCES

- *ARTIFICIAL INTELLIGENCE : RICH AND KNIGHT (TMH PUBLICATIONS*)

- *www.csee.umbc.edu/slides/ Artificial Intelligence/3-search-aiF16.pdf*