Page No.
Date

Assignment No. 1

Aim:

Study assignment : Compare different algorithm and evaluate their performance e.g depth - first search (DFS) to heristic algorithms such as Monte Carlo Tree Search (MCTS) since all of studied algorithms coverage to a solution from a solvable is to be measured by how quickly a solution was reached and how many nodes were traversed until a solution was reached.

Objective:

To compare DFS algorithm to monte carlo tree search (MCTS) algorithm.

Theory : Depth first search (DFS)

The DFS algo is a recursive algorithm that uses the idea of back-tracking. It involves exhaustive searches of all the nodes of going ahead, if possible, else by backtracking. Here, the word backtracking means that when you are moving forward and there are no more nodes along current path, move backwards to find nodes to traverse, in data.

Pseudo Code :

```
DFS ( G, v)
    u. visited = true
    for each v ∈ G.adj [u]
        if v visited = false
            DFS (G, v)

init () {
    for each u ∈ G
        u. visited = false
    for each u ∈ G
        DFS (G, u)
}
```

Complexity :

Time Complexity → O(V+E)
V = no. of nodes
E = no of edges
Space Complexity → O(V)

Monte Carlo Tree Search (MCTS)

In MCTS, nodes are building blocks of search tree It combines generality of random simulation with precision of tree seach.

1. Selection :

Starting at root node R, recursively select optimal child nodes until a leaf node L is reached. Node selection during tree descent is achieved by choosing node that maximises some quantity.

$$\left[s_i = x_i + c \times \sqrt{\frac{\ln N}{n_i}} \right]$$

2. Expansion :

If L is not terminal node then create one or more child nodes and select one C.

3. Simulation :

Run a simulated playout from C until a result is achieved.

4. Backpropogation :

Update the current move sequence with simulation result.

Pseudo Code :

```
def mcts(root):
    while resources_left(time comp power)
        leaf = traverse(root)
        sim_result = rollout(leaf)
        backprop(leaf sim_result)
    result best_child(root).
```

Function for node traversal.

```
def traverse (node) :
    while fully - expanded (node)
        node = best_ (node)
```

Function for result of simulation

```
def rollout (node) :
    while non_terminal (node)
        node = rollout_policy (node)
        return :result (node)

def rollout_policy (node)
    return pick_random (node, children)

def best_child (node) :
    pick child with highest no. of
    visits.
```

T.C. - $O(mKI/c)$
I = no. of iteration
c = no. of cores

S.C. - $O(mK)$
m = no. of ch. node
K = no. of simulation of child.

Conclusion :

Thus we have studied performance of DFS and MCTS algorithms.

(c)