### **Department of Computer Science and Engineering (Data Science)**

Subject: Artificial Intelligence (DJ19DSC502)

AY: 2023-24

**Experiment 10** 

(Planning)

60009210105

**Amitesh Sawarkar** 

D - 12

Aim: Implement a plan using AO\*.

#### Theory:

The Depth-first search and Breadth-first search given earlier for OR trees or graphs can be easily adopted by AND-OR graph. The main difference lies in the way termination conditions are determined since all goals following an AND node must be realized; whereas a single goal node following an OR node will do. So for this purpose, we are using AO\* algorithm. Like A\* algorithm here we will use two arrays and one heuristic function.

**OPEN:** It contains the nodes that have been traversed but yet not been marked solvable or unsolvable.

**CLOSE:** It contains the nodes that have already been processed.

### **AO\* Search Algorithm**

Step 1: Place the starting node into OPEN.

Step 2: Compute the most promising solution tree say T0.

Step 3: Select a node n that is both on OPEN and a member of T0. Remove it from OPEN and place it in CLOSE

Step 4: If n is the terminal goal node then leveled n as solved and leveled all the ancestors of n as solved. If the starting node is marked as solved then success and exit.

## **Department of Computer Science and Engineering (Data Science)**

Step 5: If n is not a solvable node, then mark n as unsolvable. If starting node is marked as unsolvable, then return failure and exit.

Step 6: Expand n. Find all its successors and find their h (n) value, push them into OPEN.

Step 7: Return to Step 2.

Step 8: Exit.

#### Lab Assignment to do:

Consider the use case of a plan to travel from Mumbai to Goa to attend a wedding at Taj Aguada. The plan needs to be decided based on the cost. You can either travel by train or bus or flight and stay in a hotel near or far to the wedding venue. The three options of the venues are Westin, Kennel Worth and Maria Rica hotels. You can choose between a two days package for stay and meal together or separately. Other option for your travel and stay will be a vanity van. There you need to decide if you want to cook or eat outside.

Implement AO\* to find the most suitable plan in terms of cost.



#### DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai) NAAC Accredited with "A" Grade (CGPA: 3.18)

```
• class Graph:
      def __init__(self, graph, heuristicNodeList, startNode):
          # INITIALIZING THE GRAH OBJECT WITH GRAPH TOPOLOGY, HEURISTIC VALUES AND START NODE
          self.graph = graph
          self.H=heuristicNodeList
          self.start=startNode
          self.parent={}
          ''' IF NODE IS EXPLORED, THEN STATUS = -1
              ELSE STATUS = NUMBER OF CHILD NODES OF v '''
          self.status={}
          self.solutionGraph={}
      def applyAOStar(self):
              APPLIES RECERSIVE AO* WITH BACKTRACKING FLAG
              IF FLAG = TRUE, UPDATE THE PREVIOUS H VALUES
              ELSE EXPLORE THE GRAPH FORWARD '''
          self.aoStar(self.start, False)
```



# DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

```
def getNeighbors(self, v):
        # GET NEIGHBOURS OF NODE v
        return self.graph.get(v,'')
   def getStatus(self,v):
        # RETURN STATUS OF NODE v
        return self.status.get(v,0)
   def setStatus(self, v, val):
        # SETS THE STATUS OF NODE v
        self.status[v]=val
   def getHeuristicNodeValue(self, n):
        # RETURNS H VALUE OF NODE v
        return self.H.get(n, 0)
def setHeuristicNodeValue(self, n, value):
  self.H[n]=value
def printSolution(self):
  print("\nFOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE:", self.start)
  print(self.solutionGraph)
```



## DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

```
def computeMinimumCostChildNodes(self, v):
       # COMPUTES THE PATH WITH MINIMUM COST
       minimumCost = 0
       minimumPath = []
       first = True
       for childNodesInTuples in self.getNeighbors(v):
            # ITERATING OVER ALL CHILD NODES
            cost=0
            nodeList=[]
            for c, weight in childNodesInTuples:
                 cost = cost + self.getHeuristicNodeValue(c) + weight
                 nodeList.append(c)
            # INITIALIZING MINIMUM COST = FIRST CHILD COST
            if first:
                 minimumCost=cost
                 minimumPath=nodeList
                 first = False
              # UPDATING MINIMUM COST WITH CURRECT MINIMUM COST
              elif minimumCost > cost:
                        minimumCost = cost
                        minimumPath = nodeList
        # RETURN MINIMUM COST WITH PATH
        return minimumCost, minimumPath
def aoStar(self, v, backTracking):
   print("HEURISTIC VALUES :", self.H)
   print("SOLUTION GRAPH:", self.solutionGraph)
print("PROCESSING NODE:", v)
   if v == self.start:
      print('START: ', v)
   print(f'STATUS OF {v} = {self.getStatus(v)}')
   if self.getStatus(v) >= 0:
      minimumCost, childNodeList = self.computeMinimumCostChildNodes(v)
      print(f'\nSELECTED PATH {childNodeList} FROM {v} WITH COST = {minimumCost}')
      self.setHeuristicNodeValue(v, minimumCost)
      self.setStatus(v,len(childNodeList))
      solved=True
```



#### DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai) NAAC Accredited with "A" Grade (CGPA: 3.18)

```
for childNode in childNodeList:
                          self.parent[childNode]=v
                          if self.getStatus(childNode)!=-1:
                                solved = False
                    # IF MINIMUM COST NODE 	extstyle{v} ARE SOLVED, THEN SET STATUS OF CURRENT NODE AS -1 (SOLVED)
                    # print('NODE V =', v, 'IS SOLVED =', solved, 'DO BACKTRACKING =', backTracking)
                    if solved==True:
                          self.setStatus(v, -1)
                          # UPDATE THE SOLUTION GRAPH WITH SOLVED NODES
                          self.solutionGraph[v] = childNodeList
                    # IF CURRENT NODE IS NOT START NODE, THEN BACKTRACK FOR H VALUE UPDATION
                    if v != self.start:
                          self.aoStar(self.parent[v], True)
                        # IF BACKTRACKING IS FALSE, THEN EXPLORE THE GRAPH AHEAD
                        if not backTracking:
                               # FOR EACH NODE IN MINIMUM COST PATH
                               for childNode in childNodeList:
                                      # SETTING THE STATUS OF CHILD NODE = 0
                                      self.setStatus(childNode,0)
                                      # EXPLORE THE MINIMUM COST CHILD NODE WITH NO BACKTRACKING
                                      self.aoStar(childNode, False)
• # HEURISTIC VALUE OF ALL THE NODES
  H = {'MUMBAI': 0,
         'TRAVEL': 0,
        'TRAVEL': 0,
'TRAIN': 2200, 'BUS': 2000, 'AIR': 7000,
'HOTEL BOOKING': 0,
'MARIA RIO': 0, 'WESTERN': 0, 'KENNEL WORTH': 0,
'STAY': 0, 'MEAL': 0, 'PACKAGE':0,
'VANITY VAN': 0,
'COOK': 2000, 'COST OF VANITY VAN': 30000, 'EAT (
        'COOK': 2000, 'COST OF VANITY VAN': 30000, 'EAT OUTSIDE': 3000}
  # DIRECTED GRAPH OF THE AND/OR TREE
       On = {
    'MuMBAI': [[('TRAVEL', 0), ('HOTEL BOOKING', 0)], [('VANITY VAN', 100)]],
    'TRAVEL': [[('TRAIN', 4090)], [('BUS', 1180)], [('AIR', 4050)]],
    'HOTEL BOOKING': [[('MARIA RIO', 600)], [('WESTERN', 1340)], [('KENNEL WORTH',
    'MARIA RIO': [[('STAY', 8000), ('MEAL', 4000)], [('PACKAGE', 40000)]],
    'WESTERN': [[('STAY', 34000), ('MEAL', 6000)], [('PACKAGE', 14000)]],
    'KENNEL WORTH': [[('STAY', 20000), ('MEAL', 6000)], [('PACKAGE', 32000)]],
    'VANITY VAN': [[('COOK', 0), ('COST OF VANITY VAN', 30000)], [('EAT OUTSIDE',
    0), ('COST OF VANITY VAN', 30000)]]
  G = Graph(graph, H, 'MUMBAI')
        print('APPLYING THE AO* ALGORTIHM:\n')
        G.applyAOStar()
        G.printSolution()
```



# DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

APPLYZING THE AD* ALGORIZIMI:
HEURISTIC VULUES ('MUMBAL': 0, 'TRAVEL': 0, 'TRAIN': 2200, 'BUS': 2000, 'AIR': 7000, 'HOTEL BOOKING': 0, 'MARIA RIO': 0, 'MESTERN': 0, 'KENNEL WORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COCK': 2000, 'COST O SOLUTION GRAPH : () PROCESSING NODE : MUMBAL START: MUMBAL START: MUMBAL
STATUS OF RUMBAI - 0
SELECTED PATH ['TRAVEL', 'HOTEL BOOKING'] FROM MUMBAI WITH COST = 0 HEURISTIC VALUES : ['MEMBAI': 0, 'TRAVEL': 0, 'TRAIN': 2200, 'BUS': 2000, 'AIR': 7000, 'HOTEL BOOKING': 0, 'MARIA RIO': 0, 'WESTERN': 0, 'KENNEL WORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, 'COST O SOLUTION GRAPH : () PROCESSING MODE: TRAVEL
STATUS OF TRAVEL - 0
SELECTED DATH ['BUS'] FROM TRAVEL WITH COST = 3180 HERRISTIC VALUES: ("MUMBAL": 0, "TRAVEL": 3180, "TRAIN": 2200, "BUS": 2000, "AIR": 7000, "HOTEL BOOKING": 0, "MARIA RIO": 0, "WESTERN": 0, "KEHNEL WORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN": 0, "COOK": 2000, "COS SOLUTION (RAMPH: () SOLUTION (RAMPH: () PROCESSIVE WOODE: MUMBAI START: MUMBAI
STATUS OF MAMBAL = 2
SELECTED DATH ['WANTIY VAN'] FROM MAMMAI WITH COST = 180 HEURISTIC VALUES : ('MAMMAI': 180, 'TRAVEL': 3180, 'TRAIN': 2200, '8US': 2800, 'AIR': 7000, 'HOTEL BOOKING': 0, 'MARIA RIO': 0, 'MESTERN': 0, 'KENNEL WORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANLITY VAN': 0, 'COOK': 2800, 'C SOLUTION GRAPH : () PROCESSING MODE: BUS
STATUS OF BUS = 0
SELECTED PAIR   ] FROM MUS WITH COST = 0 HENRISTIC VALUES : ("MMARIA RIO": 0, "MARIA RIO": 0, "MESTERN": 0, "KENNEL WORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN": 0, "COOK": 2000, "COS SOLUTION GRAPH : ("MIN": []) PROCESSING MODE: TRAVEL  STATUS OF TRAVEL = 1
SELECTED DATH ["BUS"] FROM TRAVEL WITH COST = 1188 HERRISTIC VALUES: ("MMRAIC": 189, "TRAVEL": 1189, "TRAIN": 2200, "BUS": 0, "AIR": 7000, "HOTEL BOOKING": 0, "MARIA RIO": 0, "MESTERN": 0, "KENNEL WORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN": 0, "COOK": 2000, "COS SOLUTION GRAPH: ("BUS": [], "TRAVEL": ["BUS"]) PROCESSIVE MODE: MARBAI START: MARBAI
STATUS OF MAMBAI = 1
SELECTED PATH ['VANITY VAN'] FROM MAMMAI MITH COST = 100 HERISIIC VALUES : ('MUMBAII: 100, 'TRAVE': 1100, 'TRAVE': 2200, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 0, 'MARIA RIO': 0, 'MESTERN': 0, 'KENNEL WORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, 'COS SOLUTION GRAPH': ("US': (), 'TRAVEL': ("BUS')) PROCESSING MODE: HOTEL BOOKING
STATUS OF HOTEL BOOKING - 0
SELECTED PATH ['MARIA RIO'] FROM HOTEL BOOKING WITH COST = 600 HEBRISIC VALUES: { "MURBAL": 1800, "TRAVEL": 1180, "TRAVEL": 2200, 'BUS': 0, "AIR': 7000, 'HOTEL BOOKING': 600, "MARIA RIO': 0, 'MESTERN': 0, 'KENNEL MORTH': 0, 'STAY': 0, 'NEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, 'CO SOLUTION GRAPH: ("BUS': 1], 'TRAVEL': ['BUS']) PROCESSING NODE: MURBAI START: HURBAI START: HURBAI
STATUS OF MAMBAI - 1
SELECTED PATH ['VANITY VAN'] FROM MAMBAI MITH COST = 100 HEURISTIC VALUES: { "MUMBAI": 100, "TRAVEL": 1180, "TRAVEL": 2000, "BUS": 0, "AIR": 7000, "HOTEL BOOKING": 600, "MARIA RIO": 0, "MESTERN": 0, "KENNEL MORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN': 0, "COOK": 2000, "CO SOLUTION GRAND - ("BUS": ], "TRAVEL": ['BUS']) PROCESSING NODE: MARIA RIO
STATUS OF MARIA RIO = 0
SELECTED PATH ['STAY', 'MEAL'] FROM MARIA RIO MITH COST = 12000 HEURISTIC VALUES : ('MUMBAI': 100, 'TRAVE': 1100, 'TRAIN': 2200, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 600, 'MARIA RIO': 12000, 'WESTERN': 0, 'KENNEL MORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, SOULTION GRAPH : ('BUS': 1), 'TRAVEL': ['BUS']} PROCESSING NODE : HOTEL BOOKING
STATUS OF HOTEL BOOKING - 1
SELECTED PATH ["MESTERN"] FROM HOTEL BOOKING WITH COST = 1340 HERRISTIC VALUES: {"MHARLIS: 100, "TAVAL": 188, "TRAIN": 2200, "BUS": 0, "AIR": 7000, "HOTEL BOOKING": 1340, "MARIA RIO": 12000, "MESTERN": 0, "KENNEL WORTH": 0, "STAV": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN": 0, "COOK": 2000 SOUTTON GRAPH: ("BUS": [], "TRAVEL": ["BUS"]) PROCESSING MODE: MUMBAI START: MUMBAI
STATUS OF MUMBAI = 1
SELECTED DATH ['WANTIY VAN'] FROM MUMBAI WITH COST = 100 HEURISTIC VALUES : {'NUMBAI': 100, 'TRAVEL': 1100, 'TRAIN': 2200, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 1340, 'MARIA RIO': 12000, 'MESTERN': 0, 'KENNEL MORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, SOLUTION GRAPH : ('BUS': [], 'TRAVEL': ['BUS']) PROCESSING MODE : STAY
STATUS OF STAY = 0
SELECTED PATH [] FROM STAY WITH COST - 0 HEURISTIC VALUES: { "NUMBALT: 180, "TRAVEL": 1180, "TRAVEL": 2000, "BUS": 0, "AIR": 7000, "HOTEL BOOKING": 1340, "MARIA RIO": 12000, "MESTERN": 0, "KENNEL MORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "WANTIY VAN": 0, "COOK": 2000, SOLUTION GRANDH: ("MS": 1], "TRAVEL": ["BUS"], "STAY": []) PROCESSING MODE: MARIA RIO
STATUS OF MARIA RIO = 2
SELECTED DATH ['STAY', 'MEAL'] FROM MARIA RIO WITH COST = 12000 HEURISTIC VALUES : ('MUMBAL': 100, 'TRAVE': 1100, 'TRAIN': 2200, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 1340, 'MARIA RIO': 12000, 'MESTERN': 0, 'KENNEL MORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, 'VANITY VAN': 0, 'COOK': 2000, SOLUTION GRAPH : ('BUS': 1), 'TRAVEL': ['BUS'], 'STAY': (]) PROCESSING MODE : HOTEL BOOKING
STATUS OF HOTEL BOOKING = 1
SELECTED PAIN ['MESTERN'] FROM HOTEL BOOKING WITH (OST - 1340 HEURISTIC VALUES : ("MBMAIL": 1800, "TRAVEL": 1800, "TRAVEL": 1800, "TRAVEL": 2000, "BUS': 0, "AIR": 7000, "HOTEL BOOKING": 1340, "MARIA RIO": 12000, "WESTERN": 0, "KENNEL MORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, "VANITY VAN": 0, "COOK": 2000, SOLUTION GRAPH: ("BUS": [], "TRAVEL": ["BUS"], "STAY": []) PROCESSING MODE: MAMBAI START: MAMBAI
STATUS OF MUMBAI - 1



# DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING



(Autonomous College Affiliated to the University of Mumbai)
NAAC Accredited with "A" Grade (CGPA: 3.18)

# **Department of Computer Science and Engineering (Data Science)**

SELECTIO PAIN ['WANII' NOW! PROM NUMBAL WITH COST = 1000 HERITSTIC VALUES ('MAMBALI' 100, TRAVEL': [1805'], "STAY': 0, "MEAL': 0, "ALR': 7000, "HOTEL BOOKING': 1340, "MARIA RIO': 12000, "MESTERN': 0, "KENNEL WORTH': 0, "STAY': 0, "MEAL': 0, "PACKAGE': 0, SOLUTION GRAPH: ("BUS'), "STAY': []} PROCESSING MODE: MEAL	↑ ↓ ⑤ 日: 'VANIII VAN:v,	CUUK : 2000
STATUS OF MEAL = 0		
SELECTED PATH [] FROM MEAL WITH COST = 0 HEURISTIC WARLES: { "MMPALT: 180, "TRAVEL': 1180, "TRAUN": 2200, "BUS': 0, "AIR": 7000, "HOTEL BOOKING": 1340, "MARIA RIO": 12000, "MESTERN": 0, "KENNEL MORTH": 0, "STAY": 0, "MEAL": 0, "PACKAGE": 0, SOUTHION GRAM! ("SUST: [], "TRAVEL": ["SUS": [], "STAY": [], "MEAL": []) PROCESSING MODE: MARIA RIO	'VANITY VAN': 0, '	COOK': 2006
STATUS OF MARIA RIO - 2		
SELECTED PATH ['STAY', 'MEAL'] FROM MARIA RIO MITH COST = 12000 HEURISTIC VALUES : { "MMPAL": 100, 'TRAVEL': 1180, 'TRAVEL': 2000, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 1340, 'MARIA RIO': 12000, 'MESTERN': 0, 'KENNEL MORTH': 0, 'STAY': 0, 'MEAL': 0, 'PACKAGE': 0, SOLUTION GRAPH : ("BUS': ], 'TRAVEL': ['BUS'], 'STAY': [], 'MEAL': [], 'MEAL': [], 'STAY', 'MEAL']) PROCESSING MODE : HOTEL BOOKING	'VANITY VAN': 0, '	COOK': 2000
STATUS OF HOTEL BOOKING = 1		
SELECTED PATH ['MESTERN'] FROM HOTEL BOOKING WITH COST - 1340 HERRISITE VALUES: ['MMPALT: 100, 'TRAVEL': 1130, 'TRANEL': 200, 'BUS': 0, 'AIR': 7000, 'HOTEL BOOKING': 1340, 'MARIA RIO': 12000, 'MESTERN': 0, 'KENNEL WORTH': 0, 'STAV': 0, 'MEAL': 0, 'PACKAGE': 0, SOLUTION GRAPH : ("BUS"], 'TRAVEL': ['BUS"], 'STAV': [], 'MEAL': [], 'MARIA RIO': ['STAV', 'MEAL']) PROCESSING MODE: MAMBAIT START: MARBAIT START: MARBAIT	'VANITY VAN': 0, '0	COOK': 2000
STATUS OF MARBAI = 1		
SELECTED PATH ['VANITY VANI'] FROM MURBAI WITH COST = 100		
FOR GRAPH SOLUTION, TRAVERSE THE GRAPH FROM THE START NODE: MAMBAI		
{'BUS': [], 'TRAVEL': ['BUS'], 'STAY': [], 'MEAL': [], 'MARIA RIO': ['STAY', 'MEAL']}		
THUS, ALGORITHM SUGGEST TO TRAVEL GOA BY BUS, AND BOOK HOTEL MARIA RIO WITH STAY	AND MEA	AL.

8