# ***Planting Agent*** Technical Report

# **Course id :AI314**

# Autonomous Multiagent Systems

# Instructor : Dr. Mohamed A. Wahby

|  |  |
| --- | --- |
| Name | ID |
| Ahmed kadry | 20180018 |
| Loai gamal | 20180206 |
| Mohamed sayed | 20180224 |
| Marina moheb | 20180208 |

7 june - Spring 2021

Robot's abilities

The main target of our agent is to sow seeds in the soil to save human energies from such an effort. This Robot should be able to do some steps to plant the soil correctly in an efficient way.

The first step is to check if the soil is suitable to be planted or not and check some other factors that affects the planting process like light and wind. In this step the robot will use its sensors to check the soil moisture, soil PH, temperature and light conditions then tells us if it’s okay to start planting or no.

The robot is able to dig inside the soil to drop the seed inside it by taking the seeds from seeds tank that found inside the robot. After dragging the seed inside the soil the robot should move a suitable distance on its wheels to sow the next seed and so on by using a sensor that detect the distance.

The robot will save a lot of time that was wasted to detect if the planting process factors would allow us sow the seeds or not. It also saves human efforts to move and dig the soil then but the seeds inside it, that is exhausting but the robot will do this rapidly.

Other Versions

* Thorvald :It is an autonomous modular robot that can be configured for mostagricultural environments. It can operate in open fields, tunnels, orchardsand greenhouses and perform tasks such as light treatment for diseasemanagement, picking fruits and vegetables, phenotyping, in-fieldtransportation, cutting grass for forage production, spraying and datacollection/crop prediction.Thorvald operates entirely on its own, using advanced navigation methodsand artificial intelligence to perform a wide variety of tasks. The robot issmall and lightweight, allowing it to perform many tasks on a farm moreeffectively and for less cost than by using tractors or manual labourers.

Rationality

# Performance measuring success

The agent will success in doing the needed task when it plants all the seeds in its seed storage take in right way with consider keeping distance between every two seeds.

# Agent prior knowledge

The agent will know the number of seeds and the area of the fertile land, but it will not know the temperature pH of the fertile land and soil moisture.

# Actions that agent can perform

The main action that the agent can perform is planting seeds in fertile land, but it also can perform some actions like knowing the temperature, the pH of the land, soil moisture, digging the land to put the seed and keep distance between every two seeds.

# Agent’s percept sequence to data

percept sequence to data

Planning

# what is our problem?

In the current generation most of the countries do not have sufficient skilled man power especially in agricultural sector and it affects the growth of developing countries. The main requirement of Automation is to reduce man power in our country. Automation saves a lot of tedious manual work and speeds up the production processes. So it is time to automate the sector to overcome this problem. So In this model the problem is in sowing the seeds.

# what is our main goal?

* We need a model that sows seeds and provides human energies in order to speed up the production process.

# planning to up coming phases:

* we have global requirement in this model is to design a robot that picks and drops seeds on the soil.
* Secondary requirements is :
* To sense the presence of soil and Divide it into squares.
* To make a hole at center of each square.
* To pick seeds from the seed-container and drop it in the hole.
* To develop a program to achieve and control all of these.

Analysis

# what is the inputs?

Take data from:

* Ultrasonic sensor :

to measure the displacement of the soil

* Soil moisture sensor:

The working of the soil moisture sensor is pretty straightforward. ... The more water in the soil means better conductivity and will result in a lower resistance. The less water in the soil means poor conductivity and will result in a higher resistance.

* pH sensors:

The availability of vitamins is as vital to plant boom as it's far to animals and livings organisms. In optimizing a plant life boom capability and yielding relatively efficient harvests, its vital to have a deep and quantitative expertise of the soil situations from which agricultural merchandise come. Using pH sensors presents crucial remarks concerning soil nutrient deficiencies or the presence of undesirable chemicals. These sensors assist clever agriculture display daily, weekly, month-to-month and annual fluctuations in soil pH and nutrient ranges to maintain to train the rural industry.

* Changing light conditions :

Sensors are measuring devices that register ambient conditions

* Temperature sensor:

an electronic device that measures the temperature of its environment and converts the input data into electronic data to record, monitor, or signal temperature changes.

# what is the outputs?

* Data from the sensors get it and move it to the program to control and make the robot take an resolution and start seeding.

PEAS

# **Performance Measure:**

* Improved efficiency in planting.
* Increased yielding and reliability in crop.
* Increased cropping frequency.
* Increased speed of seed planting.
* Seed planting accuracy.
* Durable and cheap as low-cost materials are used.
* Less maintenance cost.
* Since seed can be poured at any required depth, the plant germination is improved.
* Dependency on labor also decreased. Also, it saves time of sowing.
* Uniform placement of seeds in row with required distance.
* Proper compaction over the seeds is provided.

# Environment

* Farmer
* Seeds
* Fertile agricultural land
* Temperature
* Sun

# Actuators

Seed storage tank: Storage device is one of the important device of the system. And is designed according to weight sustained by the robot as well as the required capacity for planting. This component is stationary. To the bottom of this tank seed sowing disc is arranged. This disc serves the function of distribution of the seeds, as for each complete rotation of the rotating wheel, only one seed falls from the tank. Also number of seeds falling from tank is varied according to requirements. This disc evenly opens the way to seed hence planting is done smoothly and accurately.

Seed sowing disc & seed bucket: Disc which is attached at the bottom of the tank allows one seed during one rotation of wheel. In the above fig seed sowing disc is also included. The buckets are screwed on the disc. These buckets are very similar to half shape of pelton buckets. As these are screwed to disc its size is varied according to diameter of the seed and required distance between the seeds.

Developed Seed Mechanism: Seed metering device meters the quantity of the seed which is going into the farm. It also maintains the required level of the sand in the tank. Mostly metering is necessary to track the amount of seed also determine the when the seed tank is again filled. It gives the length or the distance which can be sowed. Thus only required seed falls for every rotation of the wheel.

Functional requirements of seed metering devices:

1. Metering of the seed should be done at a required rate. (e.g. kg/ha or seeds/meter of row length).

2. Metering should be accurate as per the requirements.

3. There should not be any damage to the seeds during metering.

# Sensors

pH Sensors in Agriculture:The availability of vitamins is as vital to plant boom as it's far to animals and livings organisms. In optimizing a plant life boom capability and yielding relatively efficient harvests, its vital to have a deep and quantitative expertise of the soil situations from which agricultural merchandise come. Using pH sensors presents crucial remarks concerning soil nutrient deficiencies or the presence of undesirable chemicals. These sensors assist clever agriculture display daily, weekly, month-to-month and annual fluctuations in soil pH and nutrient ranges to maintain to train the rural industry.

Environment type

* Fully observable because the environment is fully accessible and it has all needed information (seeds, fertile land, and all sensors I want to use)
* Deterministic certainty because the change depends on the agent action either plant the seed or refuse to plant it because the environment is inappropriate.
* Episodic because the action does not depend on the previous one.
* Dynamic because the environment maybe changes like temperature, pH of the fertile land and Soil moisture.
* Discrete because we have a limited number of distinct percepts (number of seeds and the area of the land)
* Single agent because the agent operating by itself in the environment.

Agent type

* **It is goal based agent because:**
* It has a goal to sows seeds only.
* The agent can use these goals with a set of actions and their predicted outcomes to see which action(s) achieve our goal(s).
* Achieving the goals can take 1 action or many actions. Search and planning are two subfields in AI devoted to finding sequences of actions to achieve an agents goals.

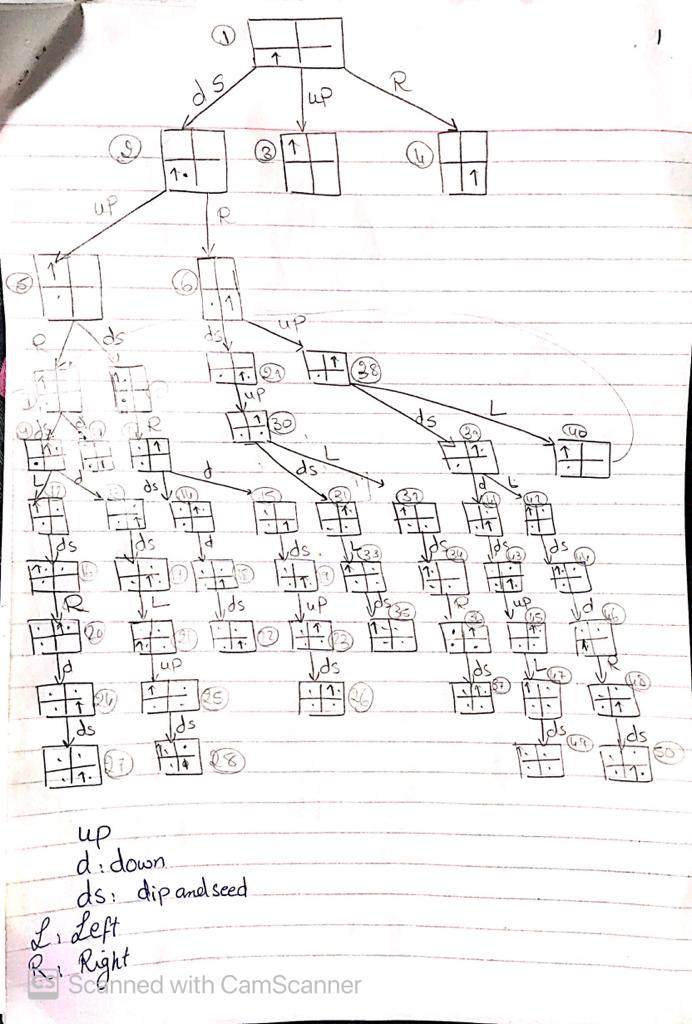
# Formulate the problem precisely:

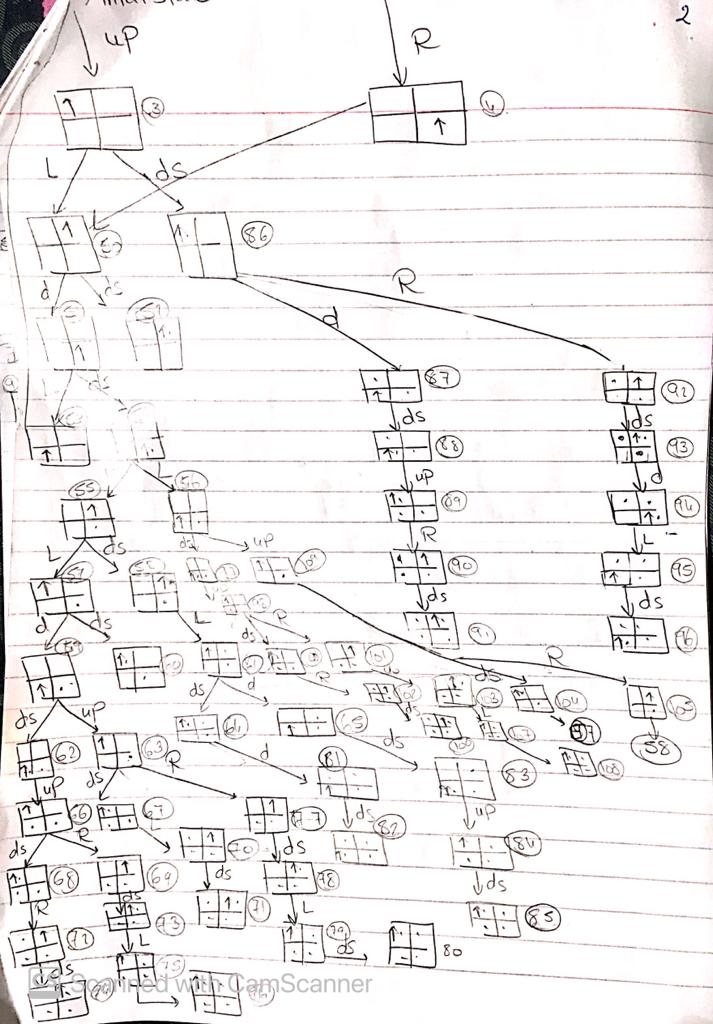
* What is our problem?
* In the current generation most of the countries do not have sufficient skilled man power especially in agricultural sector and it affects the growth of developing countries. The main requirement of Automation is to reduce man power in our country. Automation saves a lot of tedious manual work and speeds up the production processes. So it is time to automate the sector to overcome this problem. So In this model the problem is in sowing the seeds.
* What is our problem type of formulation ?
* An incremental formulation : that involves operators that augment the state description, starting with an empty state.

# Formulation

* States:
* Position of the robot.
* Bucket of the seeds is empty or not.
* The soil seeded or not.
* Initial state:
* Start of line.
* Bucket of seeds is not empty.
* The soil not seeded.
* Actions:
* Move forward on the line.
* Fill the bucket of the seeds if it empty.
* Dig the soil to make it okay for seeding.
* Seeded the soil by the seeds.
* Transition model:
* When it complete seeding in the soil on the hole line.
* Goal test:
* End of the line.
* Bucket of seeds is not empty.
* The soil is seeded.

# Diagram of the complete state space:





# The problem using two different search algorithms:

1. BFS :

BFS uses a queue to save the nodes that will be searched

queue nodes

Searched=[] // the states we have already visited

Def search(state):

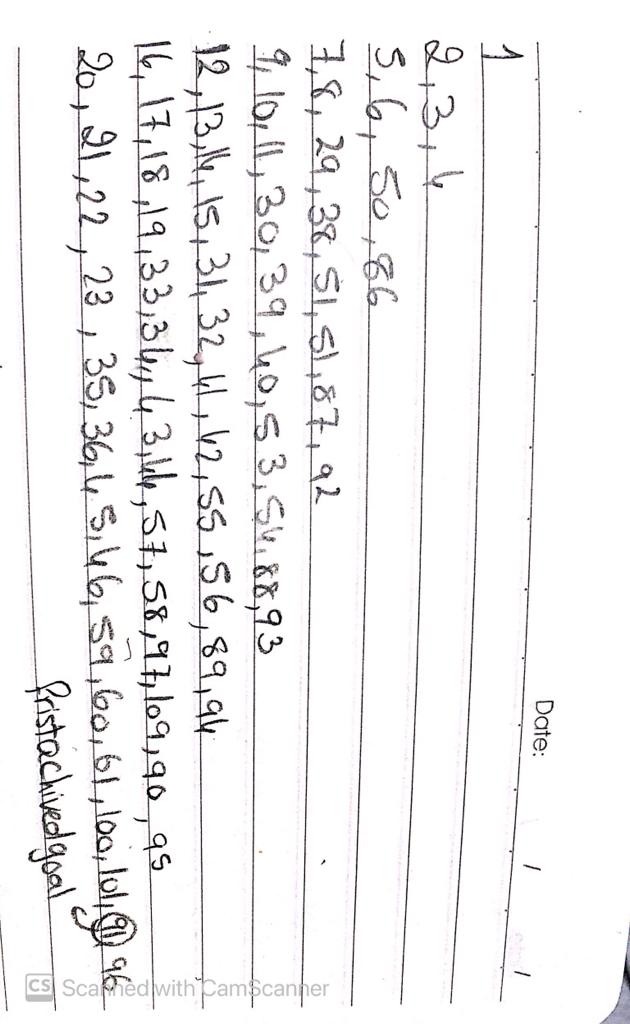
If state == goal :

Return searched

Elif state != goal and state not in sarched:

nodes.append(resultingstate(state,action))

Search(nodes[0])



States searched :

* Number of iterations: 67
* Time complexity: O(Time unit \* number of iterations=67)
* Space complexity : O(7\* nodes in each level)

2-) DFS:

DFS uses a stack to save the nodes that will be searched

Stack nodes

Searched=[] // the states we have already visited

Def search(state):

If state == goal :

Return searched

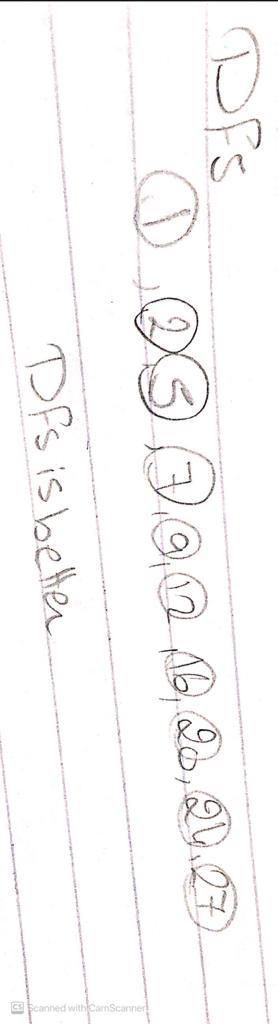
Elif state != goal and state not in sarched:

nodes.append(resultingstate(state,action))

Searched.append(state)

Search(nodes.pop(len(nodes)-1))

* States searched:



* Number of iterations: 10
* Time complexity : O (time unit \* number of iterations=10)
* Space complexity: O(10) # the maximum height of the tree

So, in our model it’s better to use depth first search because there is multiple optimal solutions and we can get them easily be searching the most left branch it will save a lot of iterations and time complexity

Code to find the solution

Class node contian : class node & state

class Node :

    def \_\_init\_\_(self,state,parent,action):

        self.state=state

        self.parent=parent

        self.action=action

class State :

    def \_\_init\_\_(self,xposition,yposition,number,seeded\_or\_not):

        self.xposition=xposition

        self.yposition=yposition

        self.no\_seeds=number # goal state should have 4 seeds but initial should have zero of it

        self.seeded=seeded\_or\_not #boolean

then , class robot :

from Node import State,Node

from Stackfrontier import Stack

class Robot:

    def \_\_init\_\_(self):

        x\_position=(input("please enter the initial horizontal position of the robot : "))

        y\_position=(input("please enter the initial vertical position of the robot : "))

        self.initial=(x\_position,y\_position,0,False)

    def get\_actions(self,Node,explored):

        Node\_state = State(Node.state[0], Node.state[1], Node.state[2], Node.state[3])

        seed\_increment=Node\_state.no\_seeds+1

        actions = [ ]

        #print((Node\_state.xposition,Node\_state.yposition))

        if (Node\_state.xposition,Node\_state.yposition) not in explored:

            Node\_state.seeded=False

        if Node\_state.xposition=="left":

           actions.append( ("right", ("right",Node\_state.yposition,Node\_state.no\_seeds,Node\_state.seeded)))

        if Node\_state.xposition=="right":

           actions.append( ("left", ("left",Node\_state.yposition,Node\_state.no\_seeds,Node\_state.seeded)))

        if Node\_state.yposition=="up":

           actions.append( ("down", (Node\_state.xposition,"down",Node\_state.no\_seeds,Node\_state.seeded)))

        if Node\_state.yposition=="down":

           actions.append( ("up", (Node\_state.xposition,"up",Node\_state.no\_seeds,Node\_state.seeded)))

        if Node\_state.seeded == False:

               actions.append(("seed", (Node\_state.xposition, Node\_state.yposition, seed\_increment, True)))

        return actions

then class solve:

from Node import Node,State

from Stackfrontier import Stack

from Robot import Robot

class solve(Robot):

        def \_\_init\_\_(self):

            super().\_\_init\_\_()

            self.solution=None

            self.num\_explored=0

            self.explored = set()

            self.position\_explored=set()

            # Initialize frontier to just the starting position

            self.start = Node(state=self.initial, parent=None, action=None)

        def solve(self):

            frontier = Stack()

            frontier.add(self.start)

            while True:

                #print("hi")

                if frontier.empty():

                    raise Exception("stack is empty")

                node = frontier.remove()

                self.num\_explored += 1

                if node.state[2]==4:

                    actions = []

                    cells = []

                    while node.parent is not None:

                        actions.append(node.action)

                        cells.append(node.state)

                        node = node.parent

                    actions.reverse()

                    cells.reverse()

                    self.solution = (actions, cells)

                    return

                self.explored.add(node.state)

                for action, state in self.get\_actions(node,explored=self.position\_explored):

                    if not frontier.contains\_state(state) and state not in self.explored:

                        child = Node(state=state, parent=node, action=action)

                        frontier.add(child)

                self.position\_explored.add((node.state[0],node.state[1]))

        def print(self):

            for i in range(len(self.solution[0])):

                print("action",i,": ",self.solution[0][i],"        state",i,": ",self.solution[1][i][:3],"\n")

then class stackfrontier :

class Stack:

    def \_\_init\_\_(self):

        self.stack=[]

    def add(self, node):

        self.stack.append(node)

    def empty(self):

        if len(self.stack)==0:

            return True

        return False

    def remove(self):

        if self.empty():

            raise Exception("empty stack")

        else:

            node=self.stack[-1]

            self.stack.pop(len(self.stack)-1)

            return node

    def contains\_state(self, state):

        return any(node.state == state for node in self.stack)

finally class draw for output:

import pygame

from solve import solve

black = [0, 0, 0]

yellow=[248,230,33]

white = [255, 255, 255]

red = [255, 0, 0]

green = [0, 255, 0]

blue=[0,0,255]

width=400

height=500

pygame.init()

pygame.display.set\_caption("robot")

pygame.font.init()

surface = pygame.display.set\_mode((400,500))

pygame.display.flip()

solve=solve()

solve.solve()

solution=solve.solution

def write\_action(action,num):

    font = pygame.font.Font('freesansbold.ttf', 32)

    text = font.render(str(num)+" "+action, True, white, black)

    textRect = text.get\_rect()

    textRect.center = (200, 450)

    pygame.draw.rect(surface, white, pygame.Rect(0, 400, 400, 100))

    return text,textRect

def seed(action,x,y):

    if action=="seed":

        if x=="left":

            xpos=100

        elif x=="right":

            xpos=300

        if y=="up":

            ypos=100

        elif y=="down":

            ypos=300

        pygame.draw.circle(surface, black,(xpos,ypos),20)

def draw\_square(action,x,y,explored):

    color=yellow

    if action != "seed":

        if x=="left":

            xpos=0

        elif x== "right":

            xpos=200

        if y=="up":

            ypos=0

        elif y=="down":

            ypos=200

        if (x,y) in explored:

            print("hi")

            color=red

            pygame.draw.rect(surface, color, pygame.Rect(xpos, ypos, 200, 200))

            seed("seed",x,y)

        else:

            pygame.draw.rect(surface, color, pygame.Rect(xpos, ypos, 200, 200))

explored=[]

runinng= True

while runinng:

    surface.fill(white)

    draw\_square("any",solve.initial[0],solve.initial[1],[])

    for event in pygame.event.get():

        if event.type == pygame.QUIT:

            pygame.quit()

        for i in range(len(solution[1])):

                state=solution[1][i]

                action=solution[0][i]

                draw\_square(action,state[0],state[1],explored)

                seed(action,state[0],state[1])

                text, textRect = write\_action(action,i)

                surface.blit(text, textRect)

                explored.append((state[0],state[1]))

                pygame.display.update()

                pygame.time.wait(2000)

        solve.print()

        pygame.quit()

References

## [1] PrasannaRaut, PradipShirwale, AbhijeetShitole “ A Survey On Smart Famer Friendly Robot Using Zigbee”, International Journal of Emerging technology and Computer Science , Volume: 01, Issue: 01, February 2016.

## [2] Calvin Hung, Juan Nieto, Zachary Taylor, James Underwood and Salah Sukkarieh, “Orchard Fruit Segmentation using Multi-spectral Feature Learning” ,IEE/RSJ International Conference on Intelligent Robot System Tokyo,Japan,3-7,November 2013.

## [3] Shrinivas R. Zanwar, R. D. Kokate, “Advanced Agriculture System”, International Journal of Robotics and Automation (IJRA), Vol. 1, No. 2, pp. 107~112 ,ISSN: 2089-4856, June 2012.

## [4] https://www.arrow.com/en/research-and-events/articles/top-5-sensors-used-in-agriculture