|  |
| --- |
| **20040121** |

**Coursework Submission Coversheet**

(individual coursework only)

**Faculty of Environment and Technology**

|  |  |
| --- | --- |
| **Field:** | Computer Science |
| **For completion by the student:** | |
| Please ensure all information is complete and correct and attach/copy this form securely to your solution.   |  | | --- | | UFCFXR-15-3 | | Autonomous Agents and Multiagent Systems | | 22 Sep TB1 | | AA-MASCW1 | | AA-MAS Portfolio Assignment |   Module code:  Module name:  Module run:  Coursework code:  Coursework title:    In submitting this form with your assignment, you make the following declaration:  **I declare that the coursework submitted is my own work and has not (either in whole or part) been submitted towards the award of any other qualification or credit either at UWE or elsewhere. I have fully attributed / referenced all sources of information used during the completion of my assignment, and I am aware that failure to do so constitutes an assessment offence.**  Name: **ABDULLA MOHAMMED ABDULMALIK**  Signature: **ABDULLA** **MOHAMMED ABDULMALIK** (Re type your name if electronically signed)  Date: \_\_**19/12/2022**\_\_\_\_\_\_\_\_\_  Time: \_\_\_\_\_**12:15 PM**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | |

**TASK 1:**

In this Assignment, I am building the autonomous system, which includes two robots with different functionalities but are dependent on each other. The two systems have triggers and its own Guards for the different states and Events that communicate with each other during the robots are in an active state. I will be following a Reactive architecture with Vertical Layering. It helps in identifying the rational behavior of the robots in the environment in this case the green area with heights varying from 1k -3k.

Following the Brook’s Subsumption Architecture, below I have based all my assumptions for the drones and robot’s multiple behaviour creating different layers of behavior.

1. All the 3 persons are located randomly on the green area
2. Movement of Drone Robot are done randomly
3. The Health’s of the each of the 3 person are randomly generated
4. The drone and First aid robots are 2 and 3 respectively.
5. The First Aid robots each run on their own batteries. And Climbing up the Terrain , drain the battery much faster as more efforts have to be put to climb up the mountain
6. Once the battery of the robots got over, it goes to its initial location, which is considered as the base location to recharge.
7. The drone are able to check the nearest First Aid robot and it battery levels. This is to make sure the First Aid Robot can complete the job of rescuing the people successfully
8. Once a successful rescue is done, the Drone and the First Aid robot return to the base safe.

Following the State Chart Diagram, below are the established Events and Triggers for the Drones and First Aid robots respectively

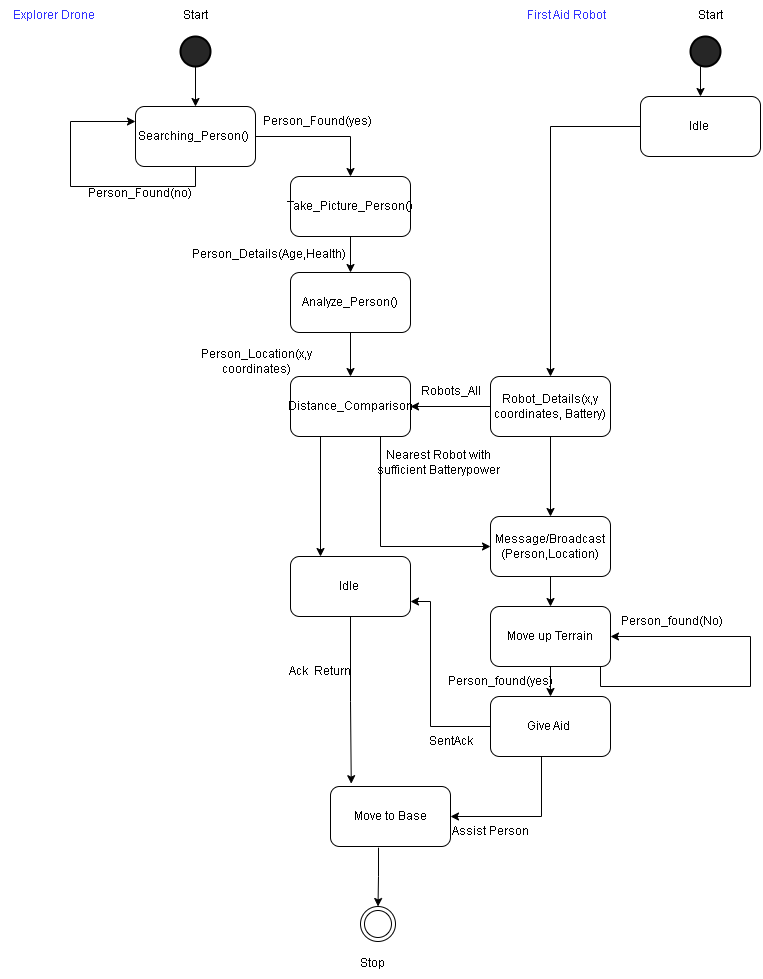
**Explorer Drones:**

1. Explore automatically for any person injured in the green area
2. The Drones are able to move in 6 Direction :Up , Down, Left, Right, Front, Back
3. Once found person is hurt, send the person’s location coordinates to the First Aid Robot
4. The Drone in general analyses the health of the person to check the nearest available First Aid Robot with enough battery to be navigated to the injured person
5. The Drone does the task of checking the battery energy of the robots and which robot best suits the work to climb up the terrain and provide supplies to the injured person.

**First Aid Robots:**

1. These Robots carry first aid like water, blanket and pain medicines
2. The Robots are able to move in 4 Direction only: Up, Down, Left, Right
3. Battery consumption is proportional to the simulation process of movement with climbing up being higher batter consumption.
4. If the charging of the battery is needed, the Robot has to go back to the base

**State Chart Diagram**



**TASK 2:**

The MESA is an Agent based Modelling framework that is currently widely used in various field like in our assignment case it is needed to deploy agents for explorer drones to navigate through the spatial grid to find persons hurt in higher altitudes and send a rescue robot agent closest to the hurt and give them first aid and escort the person back to the base. All the inbuilt function are imported from MESA to make this python code work. I have considered these agent systems as part of the Object-Oriented code language. This a dynamic evolving system, that receives and pushes information between the two types of agents.

Below is the implementation code in python documents that have been attached,

1. Mountain.py

// This contains the import libraries for MESA framework and Random library for random numbers. This is used to define the Green Area

import mesa

import random

####### Start MESA #########

high\_list=["K1","K2","K3"]

class MountainAgent(mesa.Agent):

    """An agent with fixed initial wealth."""

    def \_\_init\_\_(self, unique\_id, model):

        super().\_\_init\_\_(unique\_id, model)

        self.location=[0,0]

        self.high=self.random.choice(high\_list)

        self.Type="Mountain"

        self.bettary=0

####### END MESA ##########

1. Person.py

//This module contains the information of the person that needed to be rescued, with criteria ranging from “Very Good’ to “Very Bad’. The MESA library has also been imported here

import mesa

import random

####### Start MESA #########

health\_condition=["Very Good","Good","Bad","Very Bad"]

class PersonAgent(mesa.Agent):

    """An agent with fixed initial wealth."""

    def \_\_init\_\_(self, unique\_id, model):

        super().\_\_init\_\_(unique\_id, model)

        self.location=[0,0]

        self.age=random.randrange(15,45) #initial range

        self.health=self.random.choice(health\_condition)

        self.Type="Person"

        self.Num=0

        self.bettary=0

1. Explore\_Robot.py

// This contains all the basic and extended function for the working of the explorer drone , that also does function calls to person.py and FirstAid\_Robot.py

import mesa

import math

from FirstAid\_Robot import \*

from Person import \*

import threading

import time

####### Start MESA #########

class ExploreAgent(mesa.Agent):

    """An agent with fixed initial wealth."""

    def \_\_init\_\_(self, unique\_id, model):

        super().\_\_init\_\_(unique\_id, model)

        self.max\_speed=500 #initial value

        self.name=""  #initial value

        self.move\_direction="front" #initial value

        self.bettary=100 #initial value

        self.isServing=False #initial value

        self.nearest=-1

        self.Type="Drone"

        self.Num=0

    def step(self):

        if(self.isServing==False):

            self.move()

    def move(self):

        if(self.bettary > 0 and self.isServing==False):

            possible\_steps = self.model.grid.get\_neighborhood(

                self.pos,

                moore=True,

                include\_center=False)

            new\_position = self.random.choice(possible\_steps)

            newX,newY=new\_position #Get New Position moved to

            self.x,self.y=self.pos #Get Old position

            self.model.grid.move\_agent(self, new\_position) #move to Position

            person=self.find\_nearest()

            if person != None:

                firstAid=self.random.choice(self.model.firstAidList)

                if firstAid != None and firstAid.bettary >=50 :

                    x,y=person.pos

                    firstAid.move(x,y)

                    print('Get to Person')

            distance= math.sqrt(math.pow((self.x - newX),2) + math.pow((self.y - newY),2))

            self.bettary -= math.floor(distance)

            if(self.bettary < 0):

                self.bettary=0

        else:

            width = self.model.grid.width

            x = width - self.Num

            y = 1

            self.model.grid.move\_agent(self, (x,y))

            t1 = threading.Thread(target=self.charging)

            t1.start()

    def charging(self):

        time.sleep(10)

        self.bettary=100

    def find\_nearest(self):

        nearest=[]

        cellmates = self.model.grid.get\_cell\_list\_contents([self.pos])

        for item in cellmates:

            if isinstance(item, PersonAgent):

                nearest.append(item)

        if len(nearest) > 1:

            other = self.random.choice(cellmates)

            self.isServing = True

            print('Found Person Number: '+other.unique\_id)

            return other

        self.isServing=False

        return None

1. FirstAid\_Robot.py

//This Module contains all the basic and extended function for the working of the First Aid Robot. The different libraries used here are Math, Time etc.

import mesa

import math

import threading

import time

from Person import \*

####### Start MESA #########

class Aid:

    water=100 #initial value

    blanket=100 #initial value

    painkillers=500 #initial value

class FirstAidAgent(mesa.Agent):

    """An agent with fixed initial wealth."""

    def \_\_init\_\_(self, unique\_id, model):

        super().\_\_init\_\_(unique\_id, model)

        self.max\_speed=500 #initial value

        self.name=""

        self.aid=Aid()

        self.bettary=100 #initial value

        self.Type="First Aid"

        self.isServing=False

        self.Num=0

    def move(self,x,y):

        if(self.bettary > 0):

            self.x,self.y=self.pos

            self.model.grid.move\_agent(self, (x,y))

            distance= math.sqrt(math.pow((self.x - x),2) + math.pow((self.y - y),2) )

            if(self.bettary < 0):

                self.bettary=0

        else:

            width = self.model.grid.width

            originX = width - self.Num

            originY = 13

            self.model.grid.move\_agent(self, (originX,originY))

            t1 = threading.Thread(target=self.charging)

            t1.start()

    def charging(self):

        time.sleep(10)

        self.bettary=100

    def find\_nearest(self):

        nearest=[]

        cellmates = self.model.grid.get\_cell\_list\_contents([self.pos])

        for item in cellmates:

                if isinstance(item, PersonAgent):

                    nearest.append(item)

        if len(nearest) > 1:

            other = self.random.choice(cellmates)

            self.isServing=True

            print('Found Person Number: '+other.unique\_id)

            return other

        self.isServing=False

        return None

1. RobotAssignment.py

//This is the overall Main function that uses libraries like Mesa, numpy, matplotlib etc. to search for the injured persons and broadcast the location coordinates of the person to the First Aid robot, so the robot can take supplies to the mountain terrain, rescue the person and then assist the person safely back to the base for further medical attention. Nd at this point the rescue robot charges its battery and then becomes online ready to rescue other persons.

from random import random

import mesa

from mesa.datacollection import DataCollector

from Explore\_Robot import \*

from FirstAid\_Robot import \*

from Mountain import \*

from Person import \*

import matplotlib.pyplot as plt

import numpy as np

def compute\_gini(model):

    agent\_health = [agent.bettary for agent in model.schedule.agents]

    x = sorted(agent\_health)

    N =len(agent\_health)

    B = sum(xi \* (N - i) for i, xi in enumerate(x)) / (N \* sum(x))

    return 1 + (1 / N) - 2 \* B

#create the Model -Grid

class Model(mesa.Model):

    #A model with some number of agents.

    def \_\_init\_\_(self, ex,fa,p, width, height):

        self.num\_explore\_robot = ex

        self.num\_firstaid\_robot = fa

        self.num\_person=p

        self.grid = mesa.space.MultiGrid(width, height, True)

        self.schedule = mesa.time.RandomActivation(self)

        self.running = True

        self.Type=""

        self.droneList = [] #list of Drones

        self.firstAidList = [] #list of FirstAid

        self.personList = []  #list of Persons

        posX=1

        posY=13

        # Create agents and place them into the grid

        for i in range(self.num\_explore\_robot):

            a = ExploreAgent("Drone\_"+str(i+1), self)

            a.Num=i+1

            self.set\_position(a,i)

            self.droneList.append(a)

            self.Type="Drone"

        for j in range(self.num\_firstaid\_robot):

            b = FirstAidAgent("FirstAid\_"+str(j+1), self)

            b.Num=j+1

            self.set\_position(b,j)

            self.firstAidList.append(b)

            self.Type="First Aid"

        for k in range(self.num\_person):

            c = PersonAgent("Person\_"+str(k+1), self)

            c.Num = k+1

            self.set\_position(c,k)

            self.personList.append(c)

            self.Type="Person"

        for t in range(25): #Print the Green Area

            d = MountainAgent("Mountain\_"+ str(t+1), self)

            self.Type="Mountain"

            self.schedule.add(d)

            if(posX >= 6):

                posX = 1

                posY -= 1

            self.grid.place\_agent(d, (posX, posY))

            posX += 1

        self.data\_collector = mesa.DataCollector(

            model\_reporters={"Gini": compute\_gini}, agent\_reporters={"Bettary": "bettary"}

        )

    def step(self):

        #Advance the model by one step.

        self.data\_collector.collect(self)

        self.schedule.step()

    def set\_position(self,d,num):

        self.schedule.add(d)

        width = self.grid.width

        if(d.Type=="Drone"):

            x = width - (num +1)

            y = 1

        elif(d.Type=="First Aid"):

            x = width - (num + 1)

            y = 13

        if(d.Type=="Person"):

             min\_w,max\_w,min\_h,max\_h=1,5,10, 13

             x = self.random.randrange(min\_w,max\_w)

             y = self.random.randrange(min\_h,max\_h)

        # Add the agent to a random grid cell

        self.grid.place\_agent(d, (x, y))

########## Visualizing Object #############

def agent\_portrayal(agent):

    portrayal = {"Filled": "true",

                 "r": 0.6}

    if agent.Type =="Mountain":

        portrayal["Color"] = "lightgreen"

        portrayal["Layer"] = 0

        portrayal["Shape"]="rect"

        portrayal["w"] = 1

        portrayal["h"] = 1

    if agent.Type =="Drone":

        portrayal["Shape"]="circle"

        portrayal["Color"] = "yellow"

        portrayal["Layer"] = 1

        portrayal["text"] = agent.bettary

        portrayal["text\_color"] = "black"

    elif agent.Type =="First Aid":

        portrayal["Shape"]="circle"

        portrayal["Color"] = "red"

        portrayal["Layer"] =2

        portrayal["text"] = agent.bettary

        portrayal["text\_color"] = "black"

    elif agent.Type =="Person":

        portrayal["Shape"]="circle"

        portrayal["Color"] = "grey"

        portrayal["Layer"] = 3

        portrayal["text"] = agent.Num

        portrayal["text\_color"] = "black"

    return portrayal

########## End visualization ##############

def main():

    #####create visualization#####

    grid = mesa.visualization.CanvasGrid(agent\_portrayal,15, 15, 500, 500)

    server = mesa.visualization.ModularServer(

        Model, [grid], "Model", {"ex": 2,"fa": 3,"p":3, "width": 15, "height": 15}

    )

    server.port = 8521  # The default

    server.launch()

    #end visualization

if \_\_name\_\_ == "\_\_main\_\_":

    main()

**TASK 3:**

In this task, I have considered two performance indicators for the Robot’s scenarios for the Jupyter Notebook Analysis. This tool is a data visualization took, that takes the input as a python code, and then plots the graph for the Data analytic part of the code. The charts are based on the below two parameters for the First aid Robot as I found it critical to the development of the python code. This is because the below two indicators cannot be put as an assumption while I was writing the code. And these two parameters are needed to make sure the continuation and the effectiveness of the program.

The two different performance indications used for analysis are:

1. How long is the response time of the First Aid robot to find the person closest to it after the drone has sent the Location coordinates of the injured person?

In this Indicator, I have been using the First Aid robot to calculate the average time taken by each of the robot once it has received a signal from the explorer drone , to gradually move up the terrain and reach the injured person. Since we have more than 1 First Aid Robot, the Analysis is a graph with the x- axis being the first Aid robot and the Y-axis is the Time taken in each case. This is important to overall evaluate the performance of the robots in effectively reaching the person as their life might be depending on the supplies provided by the robot.

1. What is the Battery Energy consumption of the First Aid robot to climb 1k, 2k and 3k up the Terrain altitude to reach the first person?

In this Indicator, it is assumed, the Explorer drone sends the signal to the nearest available First Aid robot, also if the robot has enough battery energy to move up the terrain and reach the person without stopping in between. The use case is assumed that more battery energy is needed to traverse up the terrain. If the levels are less than 20% battery energy, that particular First Aid robot will not be selected, and the next First Aid robot is selected for the rescue, so the robot is successfully in delivering and assisting the injured person down the terrain and safely back to the base. Once the base is reached, both the First Aid robot and the Explorer drone recharges its battery at the base. In this case for the Jupyter notebook analysis, the chart contributes to the performance parameter of the robots, with the X-axis being the First Aid robots and the Y-axis being the Battery Energy performance. This Indicator helps in finding out how much energy is spent for recusing one person that is injured and for the future to deploy that many needed First Aid Robots.

**TASK 4:**

For this part of the Task, I have chosen the Monte Carlo Reinforcement Algorithm. The reason for choosing this algorithm is the simplicity in determining and giving a forecast with the repetition of the simulation. The outcome is all the possible outcomes with its respective underlying risk factors. This also means the output determines the quality of the input. Monte Carlo is about model free environment policy where we take the average of the outcome and repeat it to get an output that is close to the actual state value , in an environment that has several unknown factors as knowledge.

Monte Carlo methodology revolves around experience.

In our example case, we will take 1 symmetrical coin with Heads and Tails , that is tossed,with the probability of getting a Head or Tail, is each 50% or 1/2 .

H ->1/2 and T->1/2

If I toss , 2 symmetrical coins at the same time, my probabilities can be either

HH, TT, HT and TH each consisting of 0.25 probability of happening.

Now considering the possibility of tossing

the coins 1000 times, to check the actual probability of achieving a head or tail.

import random

import numpy as np

import matplotlib.pyplot as plt

#0--->Heads

#1--->Tails

def coin\_flip():

return random.randint(0,1)

coin\_flip()

list1=[]

def monte\_carlo(n):

results=0

for i in range(n):

flip\_result=coin\_flip()

results=results+flip\_result

prob\_value=results/(i+1)

list1.append(prob\_value)

plt.axhline(y=0.5,color='b',linestyle='-')

plt.xlabel("ITERATIONS")

plt.ylable("PROBABILITY")

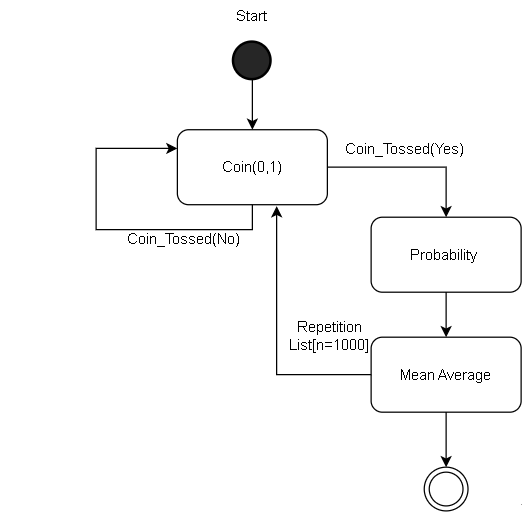
plt.plot(list1)

return results/n

answer =monte\_carlo(1000)

print("final answer:",answer)

State Chart Diagram



**CONCLUSION:**

In this scenario, I was able to understand the best fitting First Aid robot and Explorer Drone that should be used in rescuing an injured person in the Mountain Terrain. The assignment also helped me visualize data that has been collected by two autonomous agents that work with each other’s assistance.

Also building the agent using Matplotlib and Mesa framework, has enhanced my skill to develop the python code using such different libraries. Even the implementation of the grid for the movement of the First Aid Robot and Explorer Drone , helped in assumption of plotting location in Latitude and Longitude.