**Virtual Initio Programming: WS23 Sample Answers**

**Exercise 1:** Note the use of print statements. This program takes a while to run and the print statements help give the programmer a sense of progress. The program will run more quickly if epsilon\_reduce is increased, but is less likely to learn the “right” algorithm as a result. That said, the problem is simple enough that quite a high value for epsilon\_reduce can probably be used if attention span is proving a problem.

import simclient.simrobot as initio

import time, random

initio.init()

actions = ['forward','backward','left','right']

def action\_reward(action\_list, default):

 action\_rewards = {}

 for i in range(0, 2):

 for j in range(0, 2):

 for k in (action\_list):

 action\_rewards[((i, j), k)] = default

 return action\_rewards

def execute\_action(action):

 if (action == "forward"):

 initio.forward(10)

 elif (action == "backward"):

 initio.reverse(10)

 elif (action == "left"):

 initio.spinLeft(10)

 elif (action == "right"):

 initio.spinRight(10)

 else:

 initio.stop()

 time.sleep(3)

def best\_action(state):

 max\_reward = 0

 for act in actions:

 if (total\_attempts[(state, act)] > 0):

 if (reward\_dictionary[(state, act)]/total\_attempts[(state, act)] > max\_reward):

 action = act

 max\_reward = reward\_dictionary[(state, act)]/total\_attempts[(state, act)]

 else:

 if (reward\_dictionary[(state, act)] > max\_reward):

 action = act

 max\_reward = reward\_dictionary[(state, act)]

 return action

reward\_dictionary = action\_reward(actions, 0)

total\_attempts = action\_reward(actions, 0)

rewards = {(1, 1):1, (1, 0):2, (0, 1):0, (0, 0):1}

epsilon = 1

epsilon\_reduce = 0.05

while (epsilon > 0):

 explore = random.random()

 state = (initio.irLeftLine(), initio.irRightLine())

 if (explore < epsilon):

 action = random.choice(actions)

 print("Random Action: " + action)

 else:

 action = best\_action(state)

 print("Best Action: " + action)

 execute\_action(action)

 reward = rewards[(initio.irLeftLine(), initio.irRightLine())]

 reward\_dictionary[(state, action)] = reward\_dictionary[(state, action)] + reward

 total\_attempts[(state, action)] = total\_attempts[(state, action)] + 1

 if (reward == 2):

 epsilon = epsilon - epsilon\_reduce

 print("New epsilon: " + str(epsilon))

initio.stop()

print(reward\_dictionary)

for key in reward\_dictionary:

 total = total\_attempts[key]

 if (total == 0):

 print("Never attempted " + str(key))

 else:

 print("Average reward for " + str(key) + " is " + str(reward\_dictionary[key]/total\_attempts[key]))



 University of Liverpool, 2020

This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).