CSSRL

December 2020

1 Toy Problem: Access Control Queuing Task

This is a decision task involving access control to a set of 10 servers. Customers of four different priorities arrive at a single queue. If given access to a server, the customers pay a reward of 1, 2, 4, or 8 to the server, depending on their priority, with higher priority customers paying more. In each time step, the customer at the head of the queue is either accepted (assigned to one of the servers) or rejected (removed from the queue, with a reward of zero). In either case, on the next time step the next customer in the queue is considered. The queue never empties, and the priorities of the customers in the queue are uniformly randomly distributed. Of course a customer cannot be served if there is no free server; the customer is always rejected in this case. Each customer becomes free after uniform(10,20) minutes. The task is to decide on each step whether to accept or reject the next customer, on the basis of his priority and the number of free servers, so as to maximize long-term reward [1].

1.0.1 Example python code with simpy

```
# Imports
import simpy
import random
# Global variables
free_servers = 10
total_reward = 0
# Radom policy
def get_random_action(state):
  req_prio, free_servers = state
  if free_servers == 0:
    return 0
  else:
    return random.randint(0,1)
# Customer generation process
def create_customer(env):
  i=0
  while True:
    i+=1
    prios = [1,2,4,8]
    prio = random.sample(prios,k=1)[0]
    processing_time = random.uniform(10,30)
    c = Customer(i, prio, processing_time)
    assert free_servers >=0
    state = (prio, free_servers)
    action = get_random_action(state)
    env.process(c.cust_process(action))
    yield env.timeout(1)


class Customer:

  def __init__(self, name, priority, processing_time):
    self.name = name
```

[1] Inspired by Sutton and Barto Ex 10.2

```
    self.priority = priority
    self.processing_time = processing_time

  def cust_process(self, action):
    # refer to global variables
    global free_servers
    global total_reward

    print(f"Customer {self.name} arrived at {env.now}")
    if action == 0:
      print(f"Customer {self.name} with prio {self.priority} was rejected, free servers: {free_servers}")
      return
    else:
      free_servers -=1
      yield env.timeout(self.processing_time)
      free_servers+=1

      # print(self.priority)
      print(f"Customer {self.name} with prio {self.priority} was processed, free servers: {free_servers}, left at {env.now}")
      total_reward+=self.priority

# initialize simulation
env = simpy.Environment()
# Start the customer generation process
env.process(create_customer(env))
# run for 1 hour
env.run(until=60)
# summarize otal reward
print(f"Total reward:{total_reward}")
```